IBM WebSphere Application Server - Express for IBM i, Version 8.0

Administering WebSphere applications

Note

Before using this information, be sure to read the general information under “Notices” on page 3583.

Compilation date: July 25, 2011

Contents

How to send your comments xv

Changes to serve you more quickly xvii

Chapter 1. Administering ActivitySessions 1
Administering applications that use ActivitySessions 1
Enabling or disabling the ActivitySession service 1
Configuring the default ActivitySession timeout for an application server 2
ActivitySession service settings 2

Chapter 2. Administering Application profiling 5
Managing application profiles 5

Chapter 3. Administering Asynchronous beans 7
Administering asynchronous beans 7
Configuring timer managers 7
Configuring work managers 10

Chapter 4. Administering the batch environment 15
Administering the batch environment 15
Job scheduler and grid endpoint considerations 15
Configuring the job scheduler 17
Configuring WebSphere grid endpoints 22
batch jobs and their environment 24
Administrative roles and privileges 33
Administrative console help 33
Custom properties at different scopes 37
Batch administrator examples 40

Chapter 5. Administering Client applications 47
Deploying client applications 47
Deploying applet client code 47
Running an ActiveX client application 48
Deploying and running a Java EE client application 51
Running the IBM Thin Client for Enterprise JavaBeans (EJB) 155
Running Java thin client applications 157
Managing resources for Java EE client applications 160
Updating data source and data source provider configurations with the Application Client Resource Configuration Tool 160
Updating URLs and URL provider configurations for application clients 160
Updating mail session configurations for application clients 161
Updating Java Message Service provider, connection factories, and destination configurations for application clients 161
Updating WebSphere MQ as a Java Message Service provider, and its JMS resource configurations, for application clients 162
Updating resource environment entry and resource environment provider configurations for application clients 162
Removing application client resources 163
clientUpgrade script 164

Chapter 6. Administering Communications Enabled Applications 167
Administering communications enabled applications 167
Configuring services for communications enabled applications 167

Configuring communications enabled applications in a cluster . . . 169

Chapter 7. Administering Data access resources . . . 171
Deploying data access applications . . . 171
Available resources . . . 173
Map data sources for all 1.x CMP beans . . . 173
Map default data sources for modules containing 1.x entity beans . . . 175
Map data sources for all 2.x CMP beans settings . . . 176
Map data sources for all 2.x CMP beans . . . 178
Installing a resource adapter archive . . . 180
Installing resource adapters embedded within applications . . . 181
Install RAR . . . 182
Deploying SQLJ applications . . . 182
Deploying SQLJ applications that use container-managed persistence (CMP) . . . 184
Deploying SQLJ applications that use bean-managed persistence, servlets, or sessions beans 186
Customizing and binding profiles for Structured Query Language in Java (SQLJ) applications 188
Using embedded SQLJ with the DB2 for z/OS Legacy driver . . . 196
Directory conventions . . . 199
Administering data access applications . . . 201
Configuring Java EE Connector connection factories in the administrative console . . . 202
Establishing custom finder SQL dynamic enhancement server-wide . . . 234
Establishing custom finder SQL dynamic enhancement on a set of beans . . . 234
CMP connection factories collection . . . 235
Configuring resource adapters . . . 237
Updating a stand-alone resource adapter archive . . . 243
Mapping resource manager connection factory references to resource factories . . . 247
Managing messages with message endpoints . . . 248
Configuring a JDBC provider and data source . . . 251
Configuring connection validation timeout . . . 335
Resource references . . . 336
Mapping-configuration alias . . . 339
Select a J2C authentication alias . . . 340
Considerations for isolated resource providers . . . 341
Implicitly set client information . . . 341
Enabling client information tracing with the administrative console . . . 343
About Apache Derby . . . 344
Managing resources through JCA lifecycle management operations . . . 345

Chapter 8. Administering Dynamic caching . . . 349
Administering the dynamic cache service . . . 349
Using the dynamic cache service . . . 349
Disabling template-based invalidations during JSP reloads . . . 377
Dynamic cache provider for the JPA 2.0 second level cache . . . 378

Chapter 9. Administering EJB applications . . . 383
Deploying EJB 3.x enterprise beans . . . 383
EJB module settings . . . 383
Directory conventions . . . 383
Deploying EJB modules . . . 385
EJB 3.0 and EJB 3.1 deployment overview . . . 386
EJBDEPLOY relationships – troubleshooting tips . . . 388
Directory conventions . . . 388
Administering entity beans . . . 389
Enterprise beans back up and recovery best practices . . . 389
Managing EJB containers . . . 390
EJB containers . . . 391

EJB container settings . . . 392
EJB container system properties . . . 393
Changing enterprise bean types to initialize at application start time using the administrative console . . . 397
Changing applications to WebSphere "version specific" setRollbackOnly behavior . . . 398
EJB cache settings . . . 400
Container interoperability . . . 400
Configuring a timer service . . . 402
Caching data for a timer service . . . 403
Configuring the timer service using scripting . . . 406
EJB timer service settings . . . 408
Managing message-driven beans . . . 411
Managing messages with message endpoints . . . 412
Managing message listener resources for message-driven beans . . . 414
Administering applications that use the Java Persistence API . . . 431
Configure JPA to work in your environment . . . 431
Configuring OpenJPA caching to improve performance . . . 447

Chapter 10. Administering Internationalization service . . . 451
Task overview: Globalizing applications . . . 451
Globalization . . . 451
Working with locales and character encodings . . . 453
Language versions offered by this product . . . 454
Globalization: Resources for learning . . . 455
Task overview: Internationalizing interface strings (localizable-text API) . . . 455
Identifying localizable text . . . 456
Creating message catalogs . . . 456
Composing language-specific strings . . . 457
Preparing the localizable-text package for deployment . . . 465
Task overview: Internationalizing application components (internationalization service) . . . 467
Internationalization service . . . 468
Assembling internationalized applications . . . 468
Using the internationalization context API . . . 473
Administering the internationalization service . . . 492

Chapter 11. Administering Mail, URLs, and other Java EE resources . . . 499
Configuring mail providers and sessions . . . 499
Mail provider collection . . . 501
Mail provider settings . . . 501
Protocol providers collection . . . 502
Protocol providers settings . . . 502
Mail session collection . . . 502
Mail session configuration settings . . . 503
Administering URLs . . . 505
URL provider collection . . . 505
URL provider settings . . . 506
URL configurations collection . . . 506
URL configuration settings . . . 507
Administering resource environment entries . . . 508
Configuring new resource environment entries to map logical environment resource names to physical names . . . 508

Chapter 12. Administering Messaging resources . . . 515
Managing messaging with the default messaging provider . . . 515
Configuring resources for the default messaging provider . . . 516
Interoperating with a WebSphere MQ network . . . 545

Enabling WebSphere Application Server Version 5.1 JMS usage of messaging resources in later versions of the product . . . 583
Configuring the messaging engine selection process for JMS applications . . . 593
Managing messages and subscriptions for default messaging JMS destinations . . . 595
Using JMS from stand-alone clients to interoperate with service integration resources . . . 596
Using JMS from a third party application server to interoperate with service integration resources 605

Chapter 13. Managing messaging with the WebSphere MQ messaging provider . . . 717
JMS provider settings . . . 717
Scope . . . 718
Name . . . 718
Description . . . 719
Classpath . . . 719
Native library path . . . 719
Update resource adapter . . . 719
External initial context factory . . . 719
External provider URL . . . 720
Disable WebSphere MQ . . . 720
Additional properties . . . 720
Resource adapter properties . . . 721
Connection pool properties . . . 722
Max connections . . . 722
Connection concurrency . . . 722
Reconnection retry count . . . 722
Reconnection retry interval . . . 722
Additional properties . . . 722
Installing WebSphere MQ to interoperate with WebSphere Application Server . . . 722
Configuring the WebSphere MQ messaging provider with native libraries information . . . 723
Maintaining the WebSphere MQ resource adapter . . . 724
Listing JMS resources for the WebSphere MQ messaging provider . . . 727
JMS providers collection . . . 728
Activation specification collection . . . 729
Connection factory collection . . . 730
Queue connection factory collection . . . 731
Topic connection factory collection . . . 732
Queue collection . . . 732
Topic collection . . . 733
Configuring JMS resources for the WebSphere MQ messaging provider . . . 734
Creating an activation specification for the WebSphere MQ messaging provider . . . 736
Configuring an activation specification for the WebSphere MQ messaging provider . . . 738
Migrating a listener port to an activation specification for use with the WebSphere MQ messaging provider . . . 758
Creating a connection factory for the WebSphere MQ messaging provider . . . 759
Configuring a unified connection factory for the WebSphere MQ messaging provider . . . 761
Configuring a queue connection factory for the WebSphere MQ messaging provider . . . 789
Configuring a topic connection factory for the WebSphere MQ messaging provider . . . 813
Configuring a queue for the WebSphere MQ messaging provider . . . 840
Configuring a topic for the WebSphere MQ messaging provider . . . 852
Configuring custom properties for WebSphere MQ messaging provider JMS resources . . . 862
WebSphere MQ messaging provider custom properties . . . 863
Configuring custom properties for WebSphere MQ messaging provider JMS resources . . . 864
Configuring properties for the WebSphere MQ resource adapter . . . 866
Configuring custom properties for the WebSphere MQ resource adapter . . . 867
Disabling WebSphere MQ functionality in WebSphere Application Server . . . 868
WMQAdminCommands command group for the AdminTask object . . . 870
createWMQActivationSpec command . . . 871

deleteWMQActivationSpec command . . . 882
listWMQActivationSpecs command . . . 883
modifyWMQActivationSpec command . . . 884
showWMQActivationSpec command . . . 886
createWMQConnectionFactory command . . . 888
deleteWMQConnectionFactory command . . . 898
listWMQConnectionFactories command . . . 899
modifyWMQConnectionFactory command . . . 900
showWMQConnectionFactory command . . . 902
createWMQTopic command . . . 904
deleteWMQTopic command . . . 908
listWMQTopics command . . . 909
modifyWMQTopic command . . . 910
showWMQTopic command . . . 911
manageWMQ command . . . 913
showWMQ command . . . 916
migrateWMQMLP command . . . 917
createWMQQueue command . . . 919
deleteWMQQueue command . . . 922
listWMQQueues command . . . 923
modifyWMQQueue command . . . 924
showWMQQueue command . . . 926
Mapping of administrative console panel names to command names and WebSphere MQ names 927

Chapter 14. Managing messaging with a third-party or (deprecated) V5 default messaging provider . . . 933
Managing messaging with a third-party JCA 1.5-compliant messaging provider . . . 933
Configuring an activation specification for a third-party JCA resource adapter . . . 934
Configuring an administered object for a third-party JCA resource adapter . . . 938
Managing messaging with a third-party non-JCA messaging provider . . . 941
Defining a third-party non-JCA messaging provider . . . 941
Listing JMS resources for a third-party non-JCA messaging provider . . . 946
Configuring JMS resources for a third-party non-JCA messaging provider . . . 952
Maintaining (deprecated) Version 5 default messaging resources . . . 959
JMS provider settings . . . 959
Listing Version 5 default messaging resources . . . 963
Configuring Version 5 default messaging resources . . . 968

Chapter 15. Managing message-driven beans . . . 993
Managing messages with message endpoints . . . 993
Managing message listener resources for message-driven beans . . . 995
Configuring the message listener service . . . 996
Administering listener ports . . . 1004

Chapter 16. Administering Naming and directory . . . 1015
Configuring namespace bindings . . . 1015
Name space binding collection . . . 1017
Specify binding type settings . . . 1017
String binding settings . . . 1019
EJB binding settings . . . 1020
CORBA object binding settings . . . 1021
Indirect lookup binding settings . . . 1022
Configuring name servers . . . 1023
Name server settings . . . 1023

Chapter 17. Administering Object pools . . . 1025

Using object pools 1025
Object pool managers 1026
Object pool managers collection 1028
Object pool service settings 1030
Object pools: Resources for learning 1031
MBeans for object pool managers and object pools 1031

Chapter 18. Administering Object Request Broker (ORB) 1033
Administering Object Request Brokers 1033
Object Request Broker service settings 1033
Object Request Broker custom properties 1036
Character code set conversion support for the Java Object Request Broker service 1045

Chapter 19. Administering OSGi Applications 1047
Updating bundle versions for an EBA asset 1048
Updating bundle versions for an EBA asset using the editAsset command. 1049
Maintaining an OSGi composition unit 1051
Updating an OSGi composition unit 1052
Adding or removing extensions for an OSGi composition unit 1055
Modifying the configuration of an OSGi composition unit 1063
Checking the bundle download status of an EBA asset. 1076
Checking the update status of an OSGi composition unit 1077
Administering bundle repositories. 1078
Moving bundles from an OSGi application to a bundle repository 1079
Administering bundles in the internal bundle repository. 1080
Administering links to external bundle repositories 1088
Interacting with the OSGi bundle cache 1095
Exporting and importing a deployment manifest file 1097
Exporting a deployment manifest 1097
Importing a deployment manifest 1100
Default messaging provider, JMS resources 1104

Chapter 20. OSGiApplicationCommands: OSGi Applications administrative commands for the AdminTask object 1105
addExternalBundleRepository command 1105
addLocalRepositoryBundle command 1106
addOSGiExtension command 1107
addOSGiExtensions command. 1108
exportDeploymentManifest command 1109
importDeploymentManifest command 1110
listExternalBundleRepositories command 1111
listLocalRepositoryBundles command 1112
listOSGiExtensions command 1113
modifyExternalBundleRepository command 1114
removeExternalBundleRepository command 1115
removeLocalRepositoryBundle command 1115
removeLocalRepositoryBundles command 1117
removeOSGiExtension command 1118
removeOSGiExtensions command 1119
showExternalBundleRepository command. 1120
showLocalRepositoryBundle command. 1121

Chapter 21. Administering Portlet applications 1123
Portlet container settings and custom properties 1123
Portlet container settings 1123
Portlet container custom properties 1123

Portlet and PortletApplication MBeans 1124

Chapter 22. Administering Scheduler service 1127
Installing default scheduler calendars 1127
Scheduler calendars 1127
Installing default scheduler calendars 1128
Example: Using default scheduler calendars 1130
Managing schedulers 1130
Managing schedulers 1130
Scheduler daemon 1130
Example: Stopping and starting scheduler daemons using Java Management Extensions API 1131
Example: Dynamically changing scheduler daemon poll intervals using Java Management Extensions API. 1131
Configuring schedulers 1132
Creating the database for schedulers 1141

Chapter 23. Administering application security 1159
Setting up, enabling and migrating security 1159
Migrating, coexisting, and interoperating – Security considerations 1159
Enabling security 1172
Configuring multiple security domains 1218
Multiple security domains. 1222
Creating new multiple security domains 1238
Deleting multiple security domains 1241
Copying multiple security domains 1241
Configuring inbound trusted realms for multiple security domains 1245
Configure security domains 1245
External realm name 1251
Trust all realms 1251
Security domains collection 1252
Authentication cache settings 1252
Authenticating users 1254
Selecting a registry or repository 1254
Selecting an authentication mechanism 1411
Integrating third-party HTTP reverse proxy servers 1433
Single sign-on for authentication 1437
Implementing single sign-on to minimize web user authentications 1441
Configuring administrative authentication 1515
Java Authentication and Authorization Service 1516
Using the Java Authentication and Authorization Service programming model for web authentication 1519
Performing identity mapping for authorization across servers in different realms 1532
Security attribute propagation 1544
Propagating security attributes among application servers. 1549
Configuring the authentication cache 1560
Configuring Common Secure Interoperability Version 2 (CSIV2) inbound and outbound communication settings 1561
Authentication protocol for EJB security 1593
Using Microsoft Active Directory for authentication 1600
Authorizing access to resources 1614
Authorization technology 1615
Authorizing access to Java EE resources using Tivoli Access Manager 1644
Authorizing access to administrative roles 1679
Fine-grained administrative security 1688
Creating a fine-grained administrative authorization group using the administrative console 1694
Editing a fine-grained administrative authorization group using the administrative console 1696

Fine-grained administrative security in heterogeneous and single-server environments 1699
Securing communications 1700
Secure communications using Secure Sockets Layer (SSL) 1700
Creating a Secure Sockets Layer configuration. 1741
Creating a CA client in SSL 1790
Deleting a CA client in SSL 1791
Viewing or modifying a CA client in SSL 1791
Creating a keystore configuration for a preexisting keystore file. 1792
Creating a self-signed certificate 1801
Creating a certificate authority request 1805
Extracting a signer certificate from a personal certificate 1818
Retrieving signers from a remote SSL port 1822
Adding a signer certificate to a keystore 1824
Adding a signer certificate to the default signers keystore 1826
Exchanging signer certificates 1828
Configuring certificate expiration monitoring 1830
Key management for cryptographic uses 1834
Creating a key set configuration 1835
Creating a key set group configuration 1840
Auditing the security infrastructure 1847
Enabling the security auditing subsystem 1848
Creating security auditing event type filters 1853
Configuring security audit subsystem failure notifications 1861
Configuring the default audit service providers for security auditing 1864
Configuring a third party audit service providers for security auditing 1868
Configuring audit event factories for security auditing 1869
Protecting your security audit data 1872
Using the audit reader 1878

Chapter 24. Administering Service integration 1883
Enabling or disabling service integration notification events 1883
Administering service integration buses 1884
Configuring buses 1884
Operating buses 1941
Managing service integration buses with administrative commands 1943
Administering messaging engines 1943
Configuring messaging engines 1944
Starting a messaging engine 1951
Stopping a messaging engine 1951
Displaying the runtime properties of a messaging engine 1952
Displaying the runtime properties of a service integration bus link 1952
Managing messaging engines with administrative commands 1953
Administering message stores 1953
Administering file stores 1953
Administering data stores 1957
Avoiding message store errors when creating a messaging engine 1968
Avoiding errors when creating a messaging engine with a file store or a data store by using the wsadmin tool 1969
Administering bus destinations. 1970
Configuring bus destinations 1970
Managing bus destinations with administrative commands 1999
Configuring message points. 1999
Managing messages on message points 2001
Administering durable subscriptions 2002
Administering mediations. 2005
Securing mediations 2005

Configuring mediations 2007
Configuring mediation points 2018
Managing mediations with administrative commands 2020
Operating mediations at mediation points 2020
Administering messages on mediation points 2023
Example: Using mediations to trace, monitor and log messages 2024

Chapter 25. Administering Session Initiation Protocol (SIP) applications 2437
Deploying SIP applications 2437
Deploying SIP applications through the console 2437
Deploying SIP applications through scripting 2438
Administering SIP applications 2439
Configuring the SIP container 2439
Configuring SIP application routers 2462
Configuring multihomed hosting 2481
Configuring multiple proxy servers using a load balancer in a multihomed environment 2484

Chapter 26. Administering Startup beans 2487
Using startup beans 2487
Enabling startup beans in the administrative console 2488
Startup beans service settings 2488

Chapter 27. Administering Transactions 2491
Administering the transaction service 2491
Configuring transaction properties for an application server 2491
Managing active and prepared transactions 2505
Managing transaction logging for optimum server availability 2510
Displaying transaction recovery audit messages 2514
Delaying the cancelling of transaction timeout alarms 2515
Removing entries from the transaction partner log 2515

Chapter 28. Administering web applications 2517
Deploying JavaServer Pages and JavaServer Faces files 2517
JSP class loading settings 2517
JavaServer Pages (JSP) runtime reloading settings 2518
JSP and JSF option settings 2523
JSP run time compilation settings 2524
Provide options to compile JavaServer Pages settings 2525
Administering web applications 2527
Modifying the default web container configuration 2527
Configuring JSP engine parameters 2556
Backing up and recovering servlets 2570
Backing up and recovering JavaServer Pages files 2571
Administering RRD applications 2572
Asynchronous request dispatching settings 2573
Asynchronous request dispatching settings 2574
Asynchronous request dispatching settings 2574
Administering RRD applications 2575
Remote request dispatcher 2575
Configuring HTTP sessions 2576
Configuring session management by level 2576
Configuring session tracking 2579
Configuring session tracking for Wireless Application Protocol (WAP) devices 2588
Configuring for database session persistence 2588
Configuring write contents 2593
Configuring write frequency 2594

Chapter 29. Administering web services 2595
Planning to use web services 2595
Deploying web services 2596
Deploying web services applications onto application servers 2596
Using a third-party JAX-WS web services engine 2601
Deploying web services client applications 2603
Making deployed web services applications available to clients 2604
Running an unmanaged web services JAX-RPC client 2618
Running an unmanaged web services JAX-WS client 2619
Testing web services-enabled clients 2621
Administering deployed web services applications 2622
Overview of service and endpoint listeners 2623
Administration of service and endpoint listeners 2624
Viewing service providers at the cell level using the administrative console 2624
Viewing service providers at the application level using the administrative console. 2626
Viewing the detail of a service provider and managing policy sets using the administrative console 2628
Managing policy sets and bindings for service providers at the application level using the administrative console 2634
Viewing WSDL documents for service providers using the administrative console 2640
Viewing service clients at the cell level using the administrative console 2641
Viewing service clients at the application level using the administrative console. 2642
Viewing detail of a service client and managing policy sets using the administrative console 2643
Managing policy sets and bindings for services references using the administrative console 2649
Managing policy sets and bindings for service clients at the application level using the administrative console 2656
Viewing web services deployment descriptors in the administrative console 2662
Configuring the scope of a JAX-RPC web services port 2663
Suppressing the compensation service. 2665
Managing policy sets using the administrative console 2666
Viewing policy sets using the administrative console. 2667
Creating policy sets using the administrative console 2668
Importing policy sets using the administrative console 2675
Modifying policy sets using the administrative console 2678
Deleting policy sets using the administrative console 2679
Defining and managing policy set bindings 2680
Attaching a policy set to a service artifact. 2710
Managing policies in a policy set using the administrative console. 2711
Exporting policy sets using the administrative console 2759
Application policy sets collection 2760
Application policy set settings 2761
Search attached applications collection 2762
Web services policy sets 2763
Overview of migrating policy sets and bindings. 2768

Chapter 30. Administering web services - bus-enabled web services 2771
Enabling web services through the service integration bus 2771
Installing and configuring the SDO repository 2772
Configuring web services for a service integration bus 2777
Administering the bus-enabled web services resources. 2789
Creating a new WS-Security binding 2815
Creating a new WS-Security configuration 2820
Passing SOAP messages with attachments through the service integration bus. 2824

Chapter 31. Administering web services - Invocation framework (WSIF) 2955
Administering WSIF. 2955
Enabling a WSIF client to invoke a web service through JMS 2955

wsif.properties file - Initial contents 2959

Chapter 32. Administering web services - Notification (WS-Notification) 2961
Using WS-Notification for publish and subscribe messaging for web services 2961
Accomplishing common WS-Notification tasks 2962
Configuring WS-Notification resources 2985

Chapter 33. Administering web services - Policy (WS-Policy) 3099
Using WS-Policy to exchange policies in a standard format 3099
Configuring a service provider to share its policy configuration 3099
Configuring the client policy to use a service provider policy 3107
Configuring security for a WS-MetadataExchange request 3115

Chapter 34. Administering web services - Reliable messaging (WS-ReliableMessaging) . . . 3117
Administering reliable web services 3117
Configuring a WS-ReliableMessaging policy set by using the administrative console 3117
Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the administrative console 3122
Configuring endpoints to only support clients that use WS-ReliableMessaging 3125
Providing transactional recoverable messaging through WS-ReliableMessaging. 3126
WS-ReliableMessaging - administrative console panels 3127

Chapter 35. Administering web services - RESTful services 3145
Planning JAX-RS web applications 3145
Planning to use JAX-RS to enable RESTful services 3145
Defining the resources in RESTful applications. 3146
Defining the URI patterns for resources in RESTful applications 3147
Defining resource methods for RESTful applications. 3149
Defining the HTTP headers and response codes for RESTful applications. 3151
Defining media types for resources in RESTful applications 3152
Defining parameters for request representations to resources in RESTful applications 3155
Defining exception mappers for resource exceptions and errors 3158
Deploying JAX-RS web applications. 3159

Chapter 36. Administering web services - Security (WS-Security) 3163
Deploying applications that use SAML 3163
Propagating SAML tokens 3163
Creating SAML attributes in SAML tokens 3167
Establishing security context for web services clients using SAML security tokens 3169
Administering Web Services Security 3171
Configuring HTTP outbound transport level security with the administrative console 3171
Configuring HTTP outbound transport level security using Java properties. 3172
Configuring HTTP basic authentication for JAX-RPC web services with the administrative console 3173
Configuring custom properties to secure web services 3174
Administering message-level security for JAX-WS web services 3187
Administering message-level security for JAX-RPC web services 3293
Enabling cryptographic keys stored in hardware devices for Web Services Security 3439
Configuring XML digital signature for Version 5.x web services with the administrative console 3441
Configuring XML encryption for Version 5.x web services with the administrative console 3458

Chapter 37. Administering web services - Transaction support (WS-Transaction) 3467
Using WS-Transaction policy to coordinate transactions or business activities for web services 3467
Configuring a JAX-WS client for WS-Transaction context 3467
Configuring a JAX-WS web service for WS-Transaction context 3468
Configuring a WS-Transaction policy set by using wsadmin scripting. 3469
Configuring Web Services Transaction support in a secure environment 3470

Configuring an intermediary node for web services transactions . . . 3471
Enabling WebSphere Application Server to use an intermediary node for web services transactions . . . 3472
Configuring a server to use business activity support . . . 3473

Chapter 38. Administering web services - Transports . . . 3475
Invoking JAX-WS web services asynchronously using the HTTP transport . . . 3475
Using the JAX-WS asynchronous response servlet . . . 3475
Using the JAX-WS asynchronous response listener . . . 3476
Invoking JAX-WS web services asynchronously using the SOAP over JMS transport . . . 3477
Using the JAX-WS JMS asynchronous response message listener . . . 3477

Chapter 39. Administering web services - UDDI registry . . . 3481
Administering the UDDI registry . . . 3481
Setting up and deploying a new UDDI registry . . . 3481
Removing a UDDI registry node . . . 3511
Reinstalling the UDDI registry application . . . 3514
Applying an upgrade to the UDDI registry . . . 3520
Configuring SOAP API and GUI services for the UDDI registry . . . 3520
Managing the UDDI registry . . . 3522
UDDI registry administrative (JMX) interface . . . 3537
User-defined value set support in the UDDI registry . . . 3549
UDDI Utility Tools . . . 3555

Chapter 40. Administering Work area . . . 3569
Managing the UserWorkArea partition . . . 3569
Managing the UserWorkArea partition . . . 3569
Accessing the UserWorkArea partition . . . 3570
Managing local work with a work area . . . 3571
Managing local work with a work area . . . 3576
Managing local work with a work area . . . 3576
Work area service settings . . . 3577
Overriding work area properties . . . 3579
retrieveAllKeys method . . . 3580

Appendix. Directory conventions . . . 3581

Notices . . . 3583

Trademarks and service marks . . . 3585

Index . . . 3587

How to send your comments

Your feedback is important in helping to provide the most accurate and highest quality information.

- To send comments on articles in the WebSphere Application Server Information Center
  1. Display the article in your Web browser and scroll to the end of the article.
  2. Click on the **Feedback** link at the bottom of the article, and a separate window containing an e-mail form appears.
  3. Fill out the e-mail form as instructed, and click on **Submit feedback** .
- To send comments on PDF books, you can e-mail your comments to: **wasdoc@us.ibm.com** or fax them to 919-254-5250.

  Be sure to include the document name and number, the WebSphere Application Server version you are using, and, if applicable, the specific page, table, or figure number on which you are commenting.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Changes to serve you more quickly

Print sections directly from the information center navigation

PDF books are provided as a convenience format for easy printing, reading, and offline use. The information center is the official delivery format for IBM WebSphere Application Server documentation. If you use the PDF books primarily for convenient printing, it is now easier to print various parts of the information center as needed, quickly and directly from the information center navigation tree.

To print a section of the information center navigation:

1. Hover your cursor over an entry in the information center navigation until the **Open Quick Menu** icon is displayed beside the entry.
2. Right-click the icon to display a menu for printing or searching your selected section of the navigation tree.
3. If you select **Print this topic and subtopics** from the menu, the selected section is launched in a separate browser window as one HTML file. The HTML file includes each of the topics in the section, with a table of contents at the top.
4. Print the HTML file.

For performance reasons, the number of topics you can print at one time is limited. You are notified if your selection contains too many topics. If the current limit is too restrictive, use the feedback link to suggest a preferable limit. The feedback link is available at the end of most information center pages.

Under construction!

The Information Development Team for IBM WebSphere Application Server is changing its PDF book delivery strategy to respond better to user needs. The intention is to deliver the content to you in PDF format more frequently. During a temporary transition phase, you might experience broken links. During the transition phase, expect the following link behavior:

- Links to Web addresses beginning with http:// work
- Links that refer to specific page numbers within the same PDF book work
- The remaining links will *not* work. You receive an error message when you click them

Thanks for your patience, in the short term, to facilitate the transition to more frequent PDF book updates.

Chapter 1. Administering ActivitySessions

This page provides a starting point for finding information about ActivitySessions, a WebSphere extension for reducing the complexity of commitment rules and limitations that are associated with one-phase commit resources.

Use ActivitySessions to extend the scope and group multiple local transactions. With this capability, you can commit these transactions based on either deployment criteria or through explicit program logic. More introduction...

Administering applications that use ActivitySessions

You can enable or disable the ActivitySession service, and configure the default ActivitySession timeout for an application server.

About this task

You can specify whether or not the ActivitySession service is started automatically for an application server.

You can configure the default ActivitySession timeout for an application server, after which any started ActivitySessions are completed automatically by the ActivitySession service.

Procedure

- Configure the default ActivitySession timeout for an application server.
- Enable or disable the ActivitySession service.

Enabling or disabling the ActivitySession service

You can specify whether or not the ActivitySession service is started automatically for an application server.

About this task

To specify whether or not the ActivitySession service is started automatically for an application server, you can use the administrative console to configure the ActivitySession **Enable service at server startup** property.

Procedure

1. In the navigation pane of the administrative console, click **Servers** > **Server Types** > **WebSphere application servers**.
2. Click the name of the application server that you want to configure. This displays the properties for the application server in the content pane.
3. Click **[Container Settings] Business Process Services** > **ActivitySession Service** The ActivitySession service properties are displayed.
4. Select or clear the **Enable service at server startup** property as needed:
   Selected
   The ActivitySession service is started when the application server is started. Applications that specify use of ActivitySessions in their deployment descriptors can run on the application server.
   Cleared
   [Default] The ActivitySession service is not started when the application server is started. Applications that specify use of ActivitySessions in their deployment descriptors cannot start on the application server.

Any attempt to start an application that uses ActivitySessions is rejected and a message issued:

```
WACS0043E: Error found starting an application.  application_name specified an
ActivitySession attribute that is not allowed when the ActivitySession service
is not enabled
```

If this happens during server startup, the server continues to start without the application.

5. Click **OK**.
6. Save your changes to the master configuration.
7. To have the changed configuration take effect, stop then restart the application server.

Configuring the default ActivitySession timeout for an application server

Use this task to configure the default ActivitySession timeout for an application server, after which any started ActivitySessions are completed automatically by the ActivitySession service.

About this task

The ActivitySession timeout is used to reset any ActivitySession whose remote client has failed to complete the ActivitySession in a timely fashion. You can configure the initial default timeout separately for each application server, and you can override the timeout programmatically by using the setSessionTimeout method of the UserActivitySession interface. If an ActivitySession that contains a transaction reaches the timeout, the transaction's timeout is accelerated so that it is timed out (and rolled back) immediately before the ActivitySession is reset.

To configure the default ActivitySession timeout for an application server, you can use the administrative console.

Procedure

1. In the navigation pane of the administrative console, click **Servers** > **Server Types** > **WebSphere application servers**.
2. Click the name of the application server that you want to configure. This displays the properties for the application server in the content pane.
3. Click **[Container Settings] Business Process Services** > **ActivitySession Service** The ActivitySession service properties are displayed.
4. Ensure that **Enable service at server startup** is selected. You must enable the service for the timeout to have an effect.
5. In the **Default timeout** field, set the default ActivitySession timeout in seconds.
   - -1 indicates that ActivitySessions never time out
   - 0 indicates that the default timeout, 300 seconds, applies
   - Other values are an integer number of seconds
6. Click **OK**.
7. Save your changes to the master configuration.
8. For the changed configuration take effect, stop then restart the application server.

ActivitySession service settings

Use this page to configure the properties of the ActivitySession service. The ActivitySession service is a unit-of-work service to coordinate one-phase resources or to extend the activation and passivation of an enterprise bean.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **[Container Settings] Business Process Services** > **ActivitySession service**.

Enable service at server startup

Specifies whether the application server attempts to start the ActivitySession service when the server next starts up.

Default
Cleared

Range

Cleared
The server does not try to start the ActivitySession service. If ActivitySessions are to be used in applications that run on this server, the system administrator must select this property then restart the server.

Selected
When the application server starts, it attempts to start the ActivitySession service automatically.

Default timeout

Specifies the default timeout for an activity session. A server automatically completes an activity session if a remote client has failed to complete the activity session within this time period.

The initial default timeout can be configured separately for each application server, and can be overridden programmatically by the UserActivitySession interface (setSessionTimeout).

Data type
Integer

Units
Seconds

Default
300 (5 minutes)

Range
-1 through 1000000000 seconds

- -1 indicates that ActivitySessions never timeout
- 0 indicates that the default timeout applies
- Other values are an integer number of seconds

Chapter 2. Administering Application profiling

This page provides a starting point for finding information about application profiling, a WebSphere extension for defining strategies to dynamically control concurrency, prefetch, and read-ahead.

Application profiling and access intent provide a flexible method to fine-tune application performance for enterprise beans without impacting source code. Different enterprise beans, and even different methods in one enterprise bean, can have their own intent to access resources. Profiling the components based on their access intent increases performance in the application server run time.

Managing application profiles

Using the administrative console, you can add tasks to or remove tasks from application profiles.

Procedure

1. Start the administrative console.
2. Select **Applications > Enterprise Applications >** *application_name* **> Application Profiles >** *profile_name* **> Tasks**.
3. On the Tasks collection page, you can add new tasks to the profile, delete tasks, edit current task settings, and so on.

   Note that, within the scope of an application, no task can be configured on more than one application profile. In such a situation, your application cannot be restarted until you correct the configuration.
4. Save your configuration.
5. Restart the application in order for your changes to take effect.

Chapter 3. Administering Asynchronous beans

This page provides a starting point for finding information about asynchronous beans.

Asynchronous beans and asynchronous scheduling facilities offer performance enhancements for resource-intensive tasks by enabling single tasks to run as multiple tasks.

Administering asynchronous beans

Configuring timer managers

A timer manager acts as a thread pool for application components that use asynchronous beans. Use the administrative console to configure timer managers. The timer manager service is enabled by default.

Before you begin

If you are not familiar with timer managers, review the conceptual section, Timer managers, in the Asynchronous beans topic.

About this task

You can define multiple timer managers for each cell. Each timer manager is bound to a unique place in Java Naming and Directory Interface (JNDI).

Important: The timer manager service is only supported from within the Enterprise Java Beans (EJB) container or web container. Looking up and using a configured timer manager from a Java Platform, Enterprise Edition (Java EE) application client container is not supported.

Procedure

1. Start the administrative console.
2. Select **Resources > Asynchronous beans > Timer managers**.
3. Specify a **Scope** value and click **New**.
4. Specify the following required properties:
   Scope The scope of the configured resource. This value indicates the location for the configuration file.
   Name The display name for the timer manager.
   JNDI Name
   The Java Naming and Directory Interface (JNDI) name for the timer manager. This name is used by asynchronous beans that must look up the timer manager. Each timer manager must have a unique JNDI name within the cell.
   Number of Timer Threads
   The maximum number of threads that are used for timers.
5. [Optional] Specify a **Description** and a **Category** for the timer manager.
6. [Optional] Select the **Service Names** (Java EE contexts) on which you want this timer manager to be made available. Any asynchronous beans that use this timer manager then inherit the selected Java EE contexts from the component that creates the bean. The list of selected services also is known as the "sticky" context policy for the timer manager. Selecting more services than required might impede performance.
7. [Optional] Select **Custom Properties > New**. Other optional fields include:
   Name lateTimerTime
   Value Number of seconds
   Description
   Specify a description

Type Select java.lang.String

The lateTimerTime custom property is the number of seconds beyond which a late-firing timer causes an informational message to be logged. The informational message is logged once per timer manager. The default value is 5 seconds and a value of 0 disables this property.

8. Save your configuration.

Results

The timer manager is now configured and ready for access by application components that must manage the start of asynchronous code.

Timer manager collection

Use this page to view the configuration properties of timer managers, which enable applications to schedule future timer notifications and to receive timer notification callbacks to application-specified listeners within a Java 2 Platform, Enterprise Edition (J2EE) environment. The timer manager binds to the Java Naming and Directory Interface (JNDI) name space.

A timer manager contains a pool of threads bound into JNDI.

To view this administrative console page, click **Resources** > **Asynchronous beans** > **Timer managers**.

Name:

Specifies the name by which the timer manager is known for administrative purposes.

Data type String

JNDI Name:

Specifies the JNDI name used to look up the timer manager in the name space.

Data type String

Scope:

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Description:

Specifies a description of this timer manager for administrative purposes.

Data type String

Category:

Specifies a string that can be used to classify or group this timer manager.

Data type String

Timer manager settings:

Use this page to modify timer manager settings. Timer managers enable applications to schedule future timer notifications and to receive timer notification callbacks to application-specified listeners within a Java Platform, Enterprise Edition (Java EE) environment. The timer manager binds to the Java Naming and Directory Interface (JNDI) name space.

A timer manager contains a pool of threads bound into JNDI.

To view this administrative console page, click **Resources** > **Asynchronous beans** > **Timer managers** *timermanager_name*.

Scope:

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Name:

Specifies the name by which the timer manager is known for administrative purposes.

Data type String

JNDI name:

Specifies the JNDI name used to look up the timer manager in the namespace.

Data type String

Description:

Specifies a description of this timer manager for administrative purposes.

Data type String

Category:

Specifies a string that can be used to classify or group this timer manager.

Data type String

Service names:

Specifies a list of services to make available to this timer manager.

Asynchronous beans can inherit Java EE context information by enabling one or more Java EE service contexts on the timer manager resource in the product administrative console or by setting the serviceNames attribute of the TimerManagerInfo configuration object. When specifying the serviceNames attribute each enabled service should be separated by a semicolon, for example, `security;UserWorkArea;com.ibm.ws.i18n`. When a Java EE service context is enabled, it propagates the context from the scheduling thread to the target thread. If not enabled, the target thread does not inherit the context of the scheduling thread and a default context is applied. Any related Java EE context that is already present on the thread is suspended before any new Java EE context is applied.

The context information of each selected service is propagated to each timer that is created using this timer manager. Selecting services that are not needed can negatively impact performance.

Work area	Use the administrative console or the `UserWorkArea` service name to enable work area partitions. When enabled, the work area context for every work area partition that exists on the scheduling thread is available on the target thread. This feature is optional.
Security	Use the administrative console or the `security` service name to enable the Java Authentication and Authorization Service (JAAS) subject. When this feature and administrative security are enabled, the JAAS subject that is present on the scheduling thread is applied to the target thread. If not enabled, the target thread is run anonymously without a JAAS subject on the thread. This feature is optional.
Internationalization	Use the administrative console or the `com.ibm.ws.i18n` service name to enable the internationalization context information. When the internationalization context and the Internationalization service is enabled, the internationalization context that exists on the scheduling thread is available on the target thread. This feature is optional.

Number of timer threads:

Specifies the maximum number of threads that are used for timers.

Data type	Integer

Configuring work managers

A work manager acts as a thread pool for application components that use asynchronous beans. Use the administrative console to configure work managers.

Before you begin

If you are not familiar with work managers, refer to the Work managers conceptual topic.

About this task

The work manager service is always enabled. In previous versions of the product, the work manager service could be disabled using the administration console or configuration service. The work manager service configuration objects are still present in the configuration service, but the enabled attribute is ignored.

You can define multiple work managers for each cell. Each work manager is bound to a unique place in the Java Naming and Directory Interface (JNDI) namespace.

Important: The work manager service is only supported from within the Enterprise Java Beans (EJB) Container or web container. Looking up and using a configured work manager from a Java Platform, Enterprise Edition (Java EE) application client container is not supported.

Procedure

1. Start the administrative console.
2. Select **Resources > Asynchronous beans > Work managers**.
3. Specify a **Scope** value and click **New**.
4. Specify the required properties for work manager settings.

Scope The scope of the configured resource. This value indicates the location for the configuration file.

Name The display name for the work manager.

JNDI Name

The Java Naming and Directory Interface (JNDI) name for the work manager. This name is used by asynchronous beans that must look up the work manager. Each work manager must have a unique JNDI name within the cell.

Number of Alarm Threads

The maximum number of threads to use for processing alarms. A single thread is used to monitor pending alarms and dispatch them. An additional pool of threads is used for dispatching the threads. All alarm managers on the asynchronous beans associated with this work manager share this set of threads. A single alarm thread pool exists for each work manager, and all of the asynchronous beans associated with the work manager share this pool of threads.

Minimum Number Of Threads

The number of threads to be kept in the thread pool, created as needed.

Maximum Number Of Threads

Note: The maximum number of threads to be created in the thread pool. The maximum number of threads can be exceeded temporarily if the **Growable** check box is selected. These additional threads are discarded when the work on the thread completes.

Thread Priority

The priority to assign to all threads in the thread pool.

Every thread has a priority. Threads with higher priority are run before threads with lower priority. For more information about how thread priorities are used, see the Javadoc for the setPriority method of the java.lang.Thread class in the Java Standard Edition specification.

5. [Optional] Specify a **Description** and a **Category** for the work manager.
6. [Optional] Select the **Service Names** (**Java** EE contexts) on which you want this work manager to be made available. Any asynchronous beans that use this work manager then inherit the selected Java EE contexts from the component that creates the bean. The list of selected services also is known as the "sticky" context policy for the work manager. Selecting more services than are required might impede performance.

   Other optional fields include:

   Work timeout

   Specifies the number of milliseconds to wait before a scheduled work object is released. If a value is not specified, then the timeout is disabled.

   Work request queue size

   Specifies the size of the work request queue. The work request queue is a buffer that holds scheduled work objects and can be a value of 1 or greater. The thread pool pulls work from this queue. If you do not specify a value or the value is 0, the queue size is managed automatically. When the queue size is managed automatically, it is computed as the larger of (*maximum_number_of_threads*) or 20. If this value computes to a zero value, a queue size of 1 is used. Large values can consume significant system resources.

   Work request queue full action

   Specifies the action taken when the thread pool is exhausted, and the work request queue is full. This action starts when you submit non-daemon work to the work manager. If set to FAIL, the work manager API methods creates an exception instead of blocking.
7. [Optional] Select **Custom Properties > New**. Other optional fields include:

   Name lateWorkTime

   Value Number of seconds

   Description

   Specify a description

   Type Select java.lang.String

The lateWorkTime custom property is the number of seconds beyond which late-starting work must cause an informational message to be logged. The informational message is logged once per work manager. The default value is 60 seconds and a value of 0 disables this property.
Name lateAlarmTime
Value Number of seconds
Description
Specify a description
Type Select java.lang.String

The lateAlarmTime custom property is the number of seconds beyond which a late-firing alarm must cause an informational message to be logged. The informational message is logged once per work manager. The default value is 5 secondsand a value of 0 disables this property.

8. Save your configuration.

Results

The work manager is now configured and ready for access by application components that must manage the start of asynchronous code.

Work manager collection

Use this page to view the collection properties of work managers, which contain a pool of threads bound into the Java Naming and Directory Interface.

To view this administrative console page, click **Resources > Asynchronous beans > Work managers**.

Name:

Specifies the name by which the work manager is known for administrative purposes.

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name used to look up the work manager in the namespace.

Data type String

Scope:

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Description:

Specifies the description of this work manager for administrative purposes.

Category:

Specifies a category name that is used to classify or group this work manager.

Work manager settings:

Use this page to modify work manager settings. Work managers contain a pool of threads that are bound into Java Naming and Directory Interface.

To view this administrative console page, click **Resources > Asynchronous beans > Work managers >** *workmanager_name*.

Scope:

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Name:

Specifies the name by which the work manager is known for administrative purposes.

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name used to look up the work manager in the namespace.

Description:

Specifies the description of this work manager for administrative purposes.

Category:

Specifies a string that you can use to classify or group this work manager.

Work timeout:

Specifies the number of milliseconds to wait before attempting to release a unit of work. The timeout interval begins when the unit of work starts, rather than when the unit of work is submitted.

Default	0
Range	0 to int.maxvalue

Work request queue size:

Specifies the size of the work request queue.

Note: The work request queue is a buffer that holds scheduled work objects and can be a value of 1 or greater. The thread pool pulls work from this queue. If you do not specify a value or the value is 0, the queue size is managed automatically. When the queue size is managed automatically, it is computed as the larger of (*maximum_number_of_threads*) or 20. Large values can consume significant system resources.

Default 0

Work request queue full action:

Specifies the action taken when the thread pool is exhausted, and the work request queue is full. This action starts when you submit non-daemon work to the work manager.

If set to FAIL, the work manager API methods creates an exception instead of blocking.

Default BLOCK
Range FAIL

Service names:

Specifies a list of services to make available to this work manager.

Asynchronous beans can inherit J2EE context information by enabling one or more J2EE service contexts on the work manager resource in the WebSphere administrative console or by setting the serviceNames attribute of the WorkManagerInfo configuration object. When specifying the serviceNames attribute each enabled service should be separated by a semicolon. For example: `security;UserWorkArea;com.ibm.ws.i18n`. When a J2EE service context is enabled, it propagates the context from the scheduling thread to the target thread. If not enabled, the target thread does not inherit the context of the scheduling thread and a default context is applied. Any related J2EE context that is already present on the thread is suspended before any new J2EE context is applied.

The context information of each selected service is propagated to each work or alarm that is created using this work manager. Selecting services that are not needed can negatively impact performance.

Application profile (deprecated) Use the administrative console or the AppProfileService service name to enable the application profile tasks. Application profile context is not supported and not available for J2EE 1.4 applications. For J2EE 1.3 applications, the application profile context is deprecated and is only available when Application Profile Service 5.x Compatibility Mode is enabled and both the scheduling thread and target thread are J2EE 1.3 applications. When enabled, all application profile tasks that are available on the scheduling thread are available on the target thread. The scheduled work that runs in a J2EE 1.4 application does not get the application profiling task of the scheduling thread. This feature is optional.

Work area Use the administrative console or the UserWorkArea service name to enable work area partitions. When enabled, the work area context for every work area partition that exists on the scheduling thread is available on the target thread. This feature is optional.

Security Use the administrative console or the security service name to enable the Java Authentication and Authorization Service (JAAS) subject. When this feature and administrative security are enabled, the JAAS subject that is present on the scheduling thread is applied to the target thread. If not enabled, the target thread is run anonymously without a JAAS subject on the thread. This feature is optional.

Internationalization Use the administrative console or the `com.ibm.ws.i18n` service name to enable the internationalization context information. When the internationalization context and the Internationalization service is enabled, the internationalization context that exists on the scheduling thread is available on the target thread. This feature is optional.

Thread pool properties:

Specifies the priority of the threads available in this work manager.

Number of alarm threads Specifies the desired maximum number of threads used for alarms.

Minimum number of threads Specifies the minimum number of threads available in this work manager.

Maximum number of threads Specifies the maximum number of threads available in this work manager.

Thread priority Specifies the priority of the threads available in this work manager.

Growable Specifies whether the number of threads in this work manager can be increased.

Chapter 4. Administering the batch environment

You can administer the batch environment using the job scheduler, which is used to submit jobs and determine where to run them, and manage batch jobs using the job management console.

In a batch environment, many applications must complete batch work that is computational and resource intensive. Batch work might take hours or even days to finish. The batch work can use large amounts of memory or processing power while it runs.

Administering the batch environment

You can configure the batch environment and manage batch jobs.

Job scheduler and grid endpoint considerations

Configuring the batch environment includes configuring the job scheduler and grid endpoints. The job scheduler accepts job submissions and determines where to run them. Configurations for the job scheduler includes the selection of the deployment target, datasource JNDI name, database schema name, and endpoint job log location to be configured for the schedule. Batch applications are hosted in grid endpoints.

Batch environment planning for transactional batch applications and compute-intensive applications

When planning your batch environment, consider certain factors that can help you design your environment to best suit your needs.

Before you build your environment, carefully consider the goals that you want to accomplish. For example, you can configure your batch environment in an existing cell or build a new cell. Also, you must decide what relational database to use, the security you need, and what your availability requirements are. The following sections contain information about each of these considerations.

New or existing cell

You can choose to configure your batch environment in an existing WebSphere® Application Server cell or you can build a new cell entirely. Your choice depends on whether you want a new environment isolated from any existing WebSphere Application Server environment, or whether you want to add the capabilities of batch to an existing environment.

On the application server nodes where you want the job scheduler and batch container functions, use the administrative console to activate the functions. No action is necessary on the deployment manager node.

Job types

There are two job types. They are hosted in the WebSphere Application Server environment.

1. Transactional batch

   Runs transactional batch applications that are written in Java and implement a WebSphere Application Server programming model. They are packaged as Enterprise Archive (EAR) files and are deployed to the batch container hosted in an application server or cluster.

   The transactional batch programming model provides a container-managed checkpoint/restart mechanism that enables batch jobs to be restarted from the last checkpoint if interrupted by a planned or unplanned outage.
2. Compute-intensive

Runs compute-intensive applications that are written in Java and implement a WebSphere Application Server programming model. They are packaged as Enterprise Archive (EAR) files and are deployed to the batch container hosted in an application server or cluster.

The compute-intensive programming model provides a lightweight execution model based on the common framework

For all batch environments, you must deploy the job scheduler on a WebSphere Application Server server or cluster. To set up an environment to host transactional batch or compute-intensive job types, you must deploy the batch container to at least one WebSphere Application Server server or cluster. The transactional batch, compute-intensive applications, or both are installed on the same WebSphere Application Server server or cluster.

Relational database

The job scheduler and batch container both require access to a relational database. The relational database used is JDBC connected. Access to the relational database is through the underlying WebSphere Application Server connection management facilities. The relational databases supported are the same as those relational databases supported by WebSphere Application Server, including DB2®, Oracle, and others.

The simple file-based Apache Derby database is automatically configured for you by default so that you can quickly get a functioning environment up and running. However, do not use the Derby database for production use. Moreover, the default Derby database does not support a clustered job scheduler, nor a clustered batch container.

A highly available environment includes both a clustered job scheduler, and one or more clustered batch containers. Clustering requires a network database. Use production grade databases such as DB2for this purpose. Network Derby works also, but lacks the robustness necessary for production purposes. Do not use the network version in production.

Note: Application JPA settings always override the settings on this page.

Security considerations

Security for the batch environment is based on the following techniques:

1. WebSphere authentication for access to job scheduler interfaces. Users defined to the active WebSphere security registry can authenticate and gain access to the Web, command line, and programmatic interfaces of the job scheduler.
2. Role-based security for permission rights to job. Authenticated users must be assigned to the appropriate roles in order to perform actions against jobs. There are two roles:
   - lrsubmitter - Users in the lrsubmitter role can submit and operate on their own jobs, but on no others.
   - lradmin - Users in the lradmin role can submit jobs and operate on their own job or the jobs of anyone else.

   These roles are assigned through the job scheduler configuration page in the administrative console.

High availability considerations

Use clustering for high availability of batch components. Deploy and operate on clusters using the job scheduler and batch container.

Use typical application clustering techniques with the job scheduler to ensure that it is highly available. The job scheduler supports multiple methods of access to its APIs: Web application, command line, web service, and Enterprise JavaBeans (EJB). Ensuring that highly available network access to a clustered job

scheduler depends on which job scheduler API access method. The batch container is made highly available by deploying it to a cluster. The job scheduler automatically recognizes the batch container is clustered and takes advantage of it to ensure a highly available execution environment for the batch jobs that run there.

Configuring the job scheduler

The job scheduler accepts job submissions and determines where to run them. As part of managing jobs, the job scheduler stores job information in an external job database. Configurations for the job scheduler includes the selection of the deployment target, data source JNDI name, database schema name, and endpoint job log location to be configured for the scheduler.

Before you begin

See the topic about creating a non-default job scheduler and grid endpoint database.

About this task

Stand-alone application servers or clusters can host the job scheduler. The first time a server or cluster is selected to host the grid scheduler, an embedded Apache Derby database is automatically created, and configured to serve as the scheduler database if the default data source JNDI name jdbc/lrsched is selected.

The job scheduler can be configured using the administrative console or by scripting. To configure the job scheduler using the scripting language, use the link to the job scheduler configuration administrative tasks provided in the related links at the bottom of this topic. To configure the job scheduler using the administrative console, see the following procedure.

Procedure

1. Choose the environment to host the job scheduler. Use a stand-alone server for test environments. The stand-alone server can use the default Derby database. Use a cluster host for production environments. Although Derby is used as the default job scheduler database, you might want to use your own database. See the topic on creating a job scheduler and grid endpoint database for more information.
2. Log on to the administrative console.
3. Expand **System Administration** > **Job Scheduler**. The job scheduler panel opens.
4. In the **Scheduler hosted by** list, select the deployment target.
5. Type the database schema name. The default is LRSSCHEMA.
6. Select the data source JNDI name from the list. If the default of jdbc/lrsched is selected, the default embedded Derby job scheduler database is created.
7. Type the directory where the job scheduler and the batch execution environment write the job logs. The default is ${GRID_JOBLOG_ROOT}/joblogs.
8. Click **OK** and save the configuration.
9. If administrative security is enabled, enable application security and secure the job scheduler. See the topic on securing the job scheduler for more information. Only authorized users who are granted the lrmonitor, lrsubmitter, and lradmin roles, or a combination of the roles, through the administrative console are allowed access to the job management console.

Creating the job scheduler and grid endpoint database

You can create a database for the job scheduler and grid endpoint if you do not use the default Apache Derby database. The job scheduler stores job information in a relational database while the grid endpoint uses the database to track the progress of a batch job.

Before you begin

When you install the product, one Derby Java Database Connectivity (JDBC) provider is created. The Derby JDBC provider contains two data sources. One is the default Derby data source, JNDI name jdbc/lrsched, that points to the default Derby job scheduler database. The other, JNDI name jdbc/pgc, is the batch execution environment data source. If you decide to use the default data source you do not need to create the job scheduler database. The default Derby database for the job scheduler is created when the job scheduler host (deployment target) is selected through the administrative console. The default Derby database for the endpoint is created when a batch application is first installed on a node. Embedded Derby databases cannot be shared by multiple processes and are unsuitable for environments where the job scheduler must move from one node to another. For example, the job scheduler must move from one node to another in high availability scenarios.

About this task

The product supports Derby, DB2, and Oracle databases. You can use the following steps to configure the job scheduler and grid endpoint database if you decide to use a database other than the Derby database. When you create the database manually, the job scheduler and grid endpoint can use the same database.

Procedure

1. Select the correct file based on the type of database that you are going to use.

   The product provides DDL files except for DB2 on the z/OS® operating system. Use the DDL files to define the job scheduler database in the `<WAS_install_root>/util/Batch` directory. The DDL files for creating the job scheduler database are named `CreateLRSCHEDTablesXxx.ddl` where *Xxx* indicates the type of database manager that the scripts are intended for. These same DDL files are used for the grid endpoint.
2. See the documentation of your database vendor for details on customizing scripts and using the database tools to run it.

What to do next

After creating the database, complete the following steps.

1. Define the XA JDBC provider for the database through the administrative console.

   Consult the JDBC provider documentation for more information about defining a new JDBC provider.
2. Create the data source using the JDBC provider through the administrative console.

   Define the data source at the cell level. Doing so guarantees that the database is available for each application server that hosts the job scheduler.
3. Verify that the database has been created by testing the connection on the data source that you created in the previous step.
4. Configure the job scheduler by selecting the JNDI name of the newly created data source in the job scheduler panel.
5. Specify the JNDI name of the data source that you created in a previous step as the value of the GRID_ENDPOINT_DATASOURCE variable.

Verifying the job scheduler installation

This topic describes how to verify that the job scheduler is installed correctly. The job scheduler is a system application and cannot be seen in the Enterprise Application panel in the administrative console.

Before you begin

Privileges for the job scheduler differ, depending on the various roles. Roles include monitor, operator, configurator, and administrator. If you are a user with either a monitor or an operator role, you can only view the job scheduler information. If you have the role of configurator or administrator, you have all the configuration privileges for the job scheduler.

Procedure

1. Verify that the job scheduler is installed correctly by restarting the application server or cluster members where the job scheduler is configured.

   If the application server or cluster members on which the job scheduler is installed have the started icon in the status field, the job scheduler is usually running. However, the job scheduler might have a problem and not start. You can verify whether the job scheduler started by checking the log files.
2. After the server is restarted, access the job management console through a web browser by typing `http://job_scheduler_server_host:grid_host/jmc`.

   The *grid_host* port is the WC_defaulthost port under the server that you chose for the job scheduler. To find the *grid_host* port, go to your server in the administrative console, expand ports, and look for WC_defaulthost.

   If you cannot access the job management console, check the appropriate log. If you specified a server in the web address, check the server log. If you specified a cluster member in the web address, check the cluster member log.

Securing the job scheduler using roles

You can secure the job scheduler by mapping users and groups to specific security roles.

Before you begin

Users who are assigned the lradmin role have the authority to perform all job scheduler application actions on all jobs regardless of job ownership, while users who are assigned with the lrsubmitter role can only act on jobs that are owned by the submitters themselves.

Note: To start lrcmd.sh | .bat on an HTTPS port, you must configure SSL on the scheduler server. Following the steps in part three of the series, location in the following DeveloperWorks topic, Build Web services with transport-level security using Rational® Application Developer V7, Part 3: Configure HTTPS. In order to access topics, you must be a registered user for DeveloperWorks. If you have not registered as a user for DeveloperWorks, follow the instructions on the IBM® registration page.

About this task

This sample task assumes that the job scheduler is configured. From the administrative console:

Procedure

1. Click **Security > Secure administration, applications, and infrastructure**.
2. Select administrative security and application security.
3. Configure User account repository by specifying one of the available realm definitions.
4. After you have configured WebSphere Application Server Security, click **Apply** to save your configuration.
5. Expand **System administration > Job scheduler > Security role to user/group mapping**.
6. Select the roles to be configured.
7. Click **Look up users** if one or more users are to be assigned the target role, or click **Look up groups** if role assignment is at the group level.
8. Select the user or group to be assigned to the target role.
9. Click **OK** and save the configuration.
10. Restart the cell.

What to do next

With security enabled, provide a valid user ID and password for job actions that are performed through the command- line interface. Submit a job action through the command-line interface with the user name and password information. See the following example:

```
<install_root>/bin/lrcmd.[bat|sh]
-cmd=<name_of_command> <command_arguments> [-host=<host> -port=<port>]
-userid=<user_ID> -password=<password>
```

where:

- `<host>` is the job scheduler server host name. If not specified, the default is `localhost`.
- `<port>` is the scheduler server HTTP (HTTPS) port. If not specified, the default is 80.

See the following example:

```
D:\IBM\WebSphere\AppServer\bin\lrcmd
-cmd=submit -xJCL=D:\IBM\WebSphere\AppServer\samples\Batch\
postingSampleXJCL.xml -port=9445 -host=wasxd01.ibm.com
-userid=mylradmin -password=w2g0u1tf
```

Job scheduler WebSphere variables:

Use WebSphere variables to modify the job scheduler configuration. You can configure the amount of time that the job scheduler waits before signaling a problem with the endpoint and waits between polling the endpoint.

GRID_ENDPOINT_MISSED_HEART_BEAT_TOLERANCE_INTERVAL:

Define this WebSphere variable to configure the amount of time in milliseconds that the job scheduler waits between polls of the endpoint heartbeat before signaling a problem.

Table 1. GRID_ENDPOINT_MISSED_HEART_BEAT_TOLERANCE_INTERVAL. The table includes the scope, valid values, and default for the WebSphere variable.

Scope	Valid values	Default
Cell, job scheduler node, or job scheduler server level	Time in milliseconds	5 minutes

GRID_ENDPOINT_HEART_BEAT_POLL_INTERVAL:

Define this WebSphere variable to configure the amount of time in milliseconds the job scheduler waits between polls of the endpoint heartbeat.

Table 2. GRID_ENDPOINT_HEART_BEAT_POLL_INTERVAL. The table includes the scope, valid values, and default for the WebSphere variable.

Scope	Valid values	Default
Cell, job scheduler node, or job scheduler server level	Time in milliseconds	1 minute

Administrative roles and privileges:

Roles and privileges vary depending on your administrative role and the component.

Administrative roles and privileges

For definitions of administrative roles in WebSphere Application Server and how to assign them, see Authorizing access to administrative roles.

Table 3. Administrative roles and privileges. The table lists each component for the graphical user interface (GUI) and what privileges the component has for the monitor, operator, configurator, and administrator privileges.

GUI	Monitor privileges	Operator privileges	Configurator privileges	Administrator privileges
Job scheduler	View the information.	View the information.	Has all privileges.	Has all privileges.

Running batch jobs under user credentials:

This topic explains how to allow batch jobs to run under credentials of the user when WebSphere security is enabled.

About this task

The RUN_JOBS_UNDER_USER_CREDENTIAL variable allows users to enable or disable batch jobs to run under credentials of the user. When the job is dispatched to the endpoint, the batch container switches the credentials of the server to the credentials of the user. The credentials of the server are in the job step thread.

Note:

RUN_JOBS_UNDER_USER_CREDENTIAL can be created at any scope level and accepts values **true** or **false**. The default is false, which means that batch jobs run under server credentials.

When Java 2 Security is enabled, your batch applications must grant the following two permissions in the WebSphere Application Server.policy file of the application:

- permission com.ibm.websphere.security.WebSphereRuntimePermission "SecOwnCredentials"
- permission com.ibm.websphere.security.WebSphereRuntimePermission "ContextManager.getServerCredential"

The following steps describe how to create the custom property to enable or disable batch jobs to run under the credentials of a user after logging on to the administrative console:

Procedure

1. Click **Environment > WebSphere Variables**
2. Select a configuration scope, then click **New**. The general properties panel opens.
3. Type **RUN_JOBS_UNDER_USER_CREDENTIAL** in the Name field.
4. Type True or False to enable or disable jobs to run under user credential.
5. Click **OK**, then click **Save**.

What to do next

Stop and start the server where the batch execution environment is installed.

Roles and privileges for securing the job scheduler:

This topic describes the lradmin and lrsubmitter roles and privileges for securing the job scheduler.

Authority for different roles

You can secure the job scheduler application by enabling global security and application security. Application security secures the job management console. The job scheduler application uses a

combination of both declarative and instance-based security approaches to secure jobs and commands, where only users who are assigned with the lradmin or lrsubmitter role have the authority to perform grid operations in a security-enabled environment.

As illustrated in the following table, users who are assigned with the lradmin role have the authority to perform all job scheduler application actions on all jobs regardless of job ownership, while users who are assigned with the lrsubmitter role can only act on jobs that are owned by the submitters themselves. The **X** character represents authority in the following table.

Table 4. Authoritative roles. The table lists client commands and indicates with an **X** *character whether the lradmin role or the lrsubmitter role have authority for those commands.*

Client commands	lradmin role	lrsubmitter role
`submit -xJCL=<file>`	**X**	**X**
`submit -job=<job name>`	**X**	**X**
`submit -job=<job name> -add or replace`	**X**	N/A This is an admin command.
`cancel -jobid=<jobid>`	**X**	**X** (only jobs owned)
`purge -jobid=<jobid>`	**X**	**X** (only jobs owned)
`output -jobid=<jobid>`	**X**	**X** (only jobs owned)
`restart -jobid=<jobid>`	**X**	**X** (only jobs owned)
`remove -job=<jobname>`	**X**	N/A This is an admin command.
`suspend -jobid=<jobid>`	**X**	**X** (only jobs owned)
`resume -jobid=<jobid>`	**X**	**X** (only jobs owned)
`status (showAll)`	**X**	N/A This is an admin command.
`status -jobid=<jobid>`	**X**	**X** (only jobs owned)
`getBatchJobRC -jobid=<jobid>`	**X**	**X** (only jobs owned)
`help`	**X**	**X**

Configuring WebSphere grid endpoints

This topic explains how to set up a WebSphere grid endpoint.

Procedure

1. Install a batch application on a server or cluster using the administrative console.
2. If the application is the first batch application installed on the server or cluster, restart the server or cluster.

Results

The WebSphere grid endpoints are automatically set up. By installing the application on the deployment target, the common batch container is automatically deployed on the server or cluster selected using the default Apache Derby data source jdbc/pgc. The default file-based Derby data source can be used only when using the batch function on a stand-alone application server. If you have a WebSphere Application Server Network Deployment environment, you must use a network database.

If you use the default Derby data source, no further action is required. If you use a different data source, complete the following steps to make the data source available to the WebSphere grid endpoints.

1. From the administrative console, go to **Environment > WebSphere Variables**.
2. Select **Cell scope** from the list.

3. Edit the GRID_ENDPOINT_DATASOURCE variable to point to the JNDI lookup name of the job scheduler data source.
4. Save your configuration.
5. Restart all endpoints.

Creating the job scheduler and grid endpoint database

You can create a database for the job scheduler and grid endpoint if you do not use the default Apache Derby database. The job scheduler stores job information in a relational database while the grid endpoint uses the database to track the progress of a batch job.

Before you begin

When you install the product, one Derby Java Database Connectivity (JDBC) provider is created. The Derby JDBC provider contains two data sources. One is the default Derby data source, JNDI name jdbc/lrsched, that points to the default Derby job scheduler database. The other, JNDI name jdbc/pgc, is the batch execution environment data source. If you decide to use the default data source you do not need to create the job scheduler database. The default Derby database for the job scheduler is created when the job scheduler host (deployment target) is selected through the administrative console. The default Derby database for the endpoint is created when a batch application is first installed on a node. Embedded Derby databases cannot be shared by multiple processes and are unsuitable for environments where the job scheduler must move from one node to another. For example, the job scheduler must move from one node to another in high availability scenarios.

About this task

The product supports Derby, DB2, and Oracle databases. You can use the following steps to configure the job scheduler and grid endpoint database if you decide to use a database other than the Derby database. When you create the database manually, the job scheduler and grid endpoint can use the same database.

Procedure

1. Select the correct file based on the type of database that you are going to use.

   The product provides DDL files except for DB2 on the z/OS operating system. Use the DDL files to define the job scheduler database in the `<WAS_install_root>/util/Batch` directory. The DDL files for creating the job scheduler database are named `CreateLRSCHEDTablesXxx.ddl` where *Xxx* indicates the type of database manager that the scripts are intended for. These same DDL files are used for the grid endpoint.
2. See the documentation of your database vendor for details on customizing scripts and using the database tools to run it.

What to do next

After creating the database, complete the following steps.

1. Define the XA JDBC provider for the database through the administrative console.

   Consult the JDBC provider documentation for more information about defining a new JDBC provider.
2. Create the data source using the JDBC provider through the administrative console.

   Define the data source at the cell level. Doing so guarantees that the database is available for each application server that hosts the job scheduler.
3. Verify that the database has been created by testing the connection on the data source that you created in the previous step.
4. Configure the job scheduler by selecting the JNDI name of the newly created data source in the job scheduler panel.
5. Specify the JNDI name of the data source that you created in a previous step as the value of the GRID_ENDPOINT_DATASOURCE variable.

Endpoint WebSphere variables

Use WebSphere variables to modify the endpoint configuration. You can do such things as enable jobs to run under user credentials and configure the schema name of the grid endpoint database.

RUN_JOBS_UNDER_USER_CREDENTIAL:

Define this WebSphere variable so that jobs can run under user credentials.

Table 5. RUN_JOBS_UNDER_USER_CREDENTIAL. The table includes the scope, valid values, and default for the WebSphere variable.

Scope	Valid values	Default
Cell, endpoint node, or endpoint server level	• true Jobs are run under user credentials • false Jobs are run under server credentials	false

GRID_ENDPOINT_HEART_BEAT_INTERVAL:

Define this WebSphere variable to configure the amount of time between heartbeat transmissions from the grid endpoint to the job scheduler.

Table 6. GRID_ENDPOINT_HEART_BEAT_INTERVAL. The table includes the scope, valid values, and default for the WebSphere variable.

Scope	Valid values	Default
Cell, endpoint node, or endpoint server level	Time in milliseconds	30 seconds

GRID_ENDPOINT_DATASOURCE:

Define this WebSphere variable to configure the grid endpoint data source Java Naming and Directory Interface (JNDI) name.

Table 7. GRID_ENDPOINT_DATASOURCE. The table includes the scope, valid values, and default for the WebSphere variable.

Scope	Valid values	Default
Cell, endpoint node, or endpoint server level	Grid endpoint data source JNDI name	jdbc/pgc

batch jobs and their environment

The product provides ways of managing the scheduling and execution control of background activities in a grid computing environment.

The various ways that you can manage your batch environment include using the job management console, analyzing job logs, specifying job classes, and by using classification rules.

Through the job management console, you can:

- Submit jobs
- Monitor job execution
- Perform operational actions against jobs

- View job logs

Job logs

A job log is a file that contains a detailed record of the execution details of a job. It is composed of both system and application messages. Job logs are stored on the endpoints where the job runs and on the application server that hosts the job scheduler.

Job logs are viewable through the job management console and from the command line.

Job management console

The job management console is a stand-alone web interface that you can use to perform job operations such as submit, monitor, schedule, and manage.

The job management console is for managing jobs. This console provides controlled access when security is enabled.

Only authorized users who are granted the lrsubmitter role, the lradmin role, or both roles through the administrative console can be allowed access to the job management console.

Through the job management console, you can:

- Submit jobs.
- Monitor job execution.
- Perform operational actions against jobs.
- View job logs.

To access the job management console:

1. Configure the job scheduler.
2. Ensure that the job scheduler is running.

   If the application server or cluster members on which the job scheduler is installed have the started icon in the status field, the job scheduler is usually running. However, the job scheduler might have a problem and not start. You can verify whether the job scheduler started by checking the log files.
3. In a browser, type the web address: `http://<job scheduler server host>:<port>/jmc`.
4. If an on-demand router (ODR) is defined in the cell, type the web address: `http://<odr host>:80/jmc`.
5. If you cannot access the job management console, check the appropriate log. If you specified a server in the web address, check the server log. If you specified a cluster member in the web address, check the cluster member log.

To access the field help for the job management console, click **?** in the upper right corner of every job management panel.

Command-line interface

The command-line interface interacts with the job scheduler to submit and manipulate a batch job. It is located in the *was_root*/bin directory as the lrcmd.sh or lrcmd.bat script and can be started from any location in the WebSphere cell.

Use the lrcmd script to perform the following commands:

Table 8. lrcmd commands. The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional Information
Display usage information for lrcmd.	None	The command displays usage information for the lrcmd command.	Example: lrcmd

Table 8. lrcmd commands (continued). The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional Information
Submit a job to the job scheduler.	-cmd=submit -xJCL=<xjcl_filename> [-host=<host>] [-port=<port>], or -cmd=submit -job=<job_name> [-startDate=<startDate> - startTime=<startTime>] [-host=<host>] [-port=<port>]	When an XML Job Control Language (xJCL) file is specified, -xJCL=<xjcl_filename> specifies the path of the xJCL to be submitted from the file system and optionally saved. Optional arguments: • Use -host=<host> as the on-demand router (ODR) host name or job scheduler server host name. If not specified, the default is localhost. • Use -port=<port> as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80. Both variations of the command return a job ID for the submitted job.	Examples: • lrcmd -cmd=submit -xJCL=myxjcl.xml -host=myhost -port=81 • lrcmd -cmd=submit -xJCL=myxjcl.xml • lrcmd -cmd=submit -job=myjob
Cancel a previously submitted job.	-cmd=cancel -jobid=<jobid> [-<host>] [-port=<port>]	This command cancels the start of a previously submitted job, or cancels the execution of a running job. Use -jobid=<jobid> as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The -cmd=status command can also be used to identify the job ID for a particular job. Optional arguments: • Use -host=<host> as the ODR host name or job scheduler server host name. If not specified, the default is localhost. • Use -port=<port> as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.	Example: lrcmd -cmd=cancel -jobid=myjob:2 -host=myLRShost -port=9083
Restart a job.	-cmd=restart -jobid=<jobid> [-host=<host>] [-port=<port>]	This command restarts the start of a job. Only jobs in restartable state can be restarted. Use -jobid=<jobid> as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The -cmd=status command can also be used to identify the job ID for a particular job. Optional arguments: • Use -host=<host> as the ODR host name or job scheduler server host name. If not specified, the default is localhost. • Use -port=<port> as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.	Only a batch job associated with batch applications can be restarted. When a batch job is canceled using the -cmd=cancel command, its state is changed to restartable. When the job is restarted, processing resumes from the last successfully committed checkpoint. Example: lrcmd -cmd=restart -jobid=myjob:2 -host=myLRShost -port=9081

Table 8. lrcmd commands (continued). The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional Information

Purge job information.	`-cmd=purge`
	`-job=<jobid>`
	`[-host=<host>] [-port=<port>]`
		This command purges job information from the job scheduler and grid endpoints.
		
		The job scheduler maintains information about a job after the job has completed. The purge command permanently deletes job information from the job scheduler and grid endpoint databases. The command also purges the job log of the job.
		
		Use `-jobid=<jobid>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the `lrcmd -cmd=submit` command that initially submitted the job. The -cmd=status command can also be used to identify the job ID for a particular job.
		
		Optional arguments:
		• Use `-host=<host>` as the ODR host name or job scheduler server host name. If not specified, the default is localhost.
		• Use `-port=<port>` as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.
			The job scheduler maintains information about a job after the job has completed. The purge command permanently deletes job information from the job scheduler and grid endpoint databases. The command also purges the job log of the job.
			
			Example:
			
			`lrcmd -cmd=purge -jobid=myjob:2`

Show the status of a batch job.	`-cmd=status`
	
	or
	
	`-cmd=status`
	`-jobid=<jobid>`
	`[-host=<host>] [-port=<port>]`
		This command displays status information about one or more jobs in the job scheduler database.
		
		Optional argument: `-job=<jobid>`, if specified, indicates that only job information for the specified job is displayed.
			Examples:
			• `lrcmd -cmd=status host=myODRHost -port=83`
			• `lrcmd -cmd=submit -xJCL=myxjcl.xml` (returns job ID LongRunningScheduler:17)
			• `lrcmd -cmd=status -jobid=LongRunningScheduler:17`

Suspend a job.	`-cmd=suspend`
	`-jobid=<jobid>`
	`-seconds=<seconds>`
	`[-host=<host>] [-port=<port>]`
		This command suspends the start of a grid batch job for the specified number of seconds. Unless manually resumed (with `lrcmd -cmd=resume`, for example), the job automatically resumes running after the specified number of seconds.
		
		Use `-jobid=<jobid>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The `-cmd=status` command can also be used to identify the job ID for a particular job.
		
		Optional arguments:
		
		Use `-seconds=<seconds>` to indicate the number of seconds that the job start is suspended. If not specified, the default value of 15 seconds is used. If `-seconds=0` is specified, the job does not start until manually resumed.
		• Use `-host=<host>` as the ODR host name or job scheduler server host name. If not specified, the default is localhost.
		• Use `-port=<port>` as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.
			Examples:
			
			`lrcmd -cmd=submit -xJCL=myxjcl.xml` (returns job ID myjob:23). After job myjob:23 has begun execution, it can be suspended for five minutes (for example), with: `lrcmd -cmd=suspend -jobid=myjob:23 -seconds=300 -port=81 -host=myODRHost`
			
			Execution of the job can be resumed before the 5 minutes expires with: `lrcmd -cmd=resume -jobid=myjob:23`

Table 8. lrcmd commands (continued). The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional Information
Resume start of a previously suspended job.	`-cmd=resume -jobid=<jobid> [-host=<host>] [-port=<port>]`	This command resumes start of a previously suspended batch job. Use `-jobid=<jobid>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The -cmd=status command can also be used to identify the job ID for a particular job.	See description of -cmd=suspend.
Display the output for a job.	`-cmd=output -jobid=<jobid> [-host=<host>] [-port=<port>]`	Displays the output generated by the job scheduler and grid endpoint during the execution of the specified job. Use `-jobid=<jobid>` as the ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The -cmd=status command can also be used to identify the job ID for a particular job.	(none)
Display the return code of a batch job.	`-cmd=getBatchJobRC -jobid=<jobid> [-host=<host>] [-port=<port>]`	Displays the overall return code produced by a grid batch job. Use `-jobid=<jobid>` as the ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The -cmd=status command can also be used to identify the job ID for a particular job. • Use `-host=<host>` as the ODR host name or job scheduler server host name. If not specified, the default is localhost. • Use `-port=<port>` as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.	(none)

Table 9. lrcmd commands. The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional information
Display usage information for lrcmd.	None	This command displays usage information for the `lrcmd` command.	Example: `lrcmd`
Stop the execution of a previously submitted job.	`-cmd=stop [-jobid=<job_id> [-host=<host>] [-port=<port>]`	This command stops the execution of a previously submitted job when a checkpoint occurs. Use `-jobid=<jobid>` as the job ID assigned to the job by the job scheduler Optional arguments: • Use `-host=<host>` as the ODR host name or job scheduler server host name. If not specified, the default is localhost. • Use `-port=<port>` as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.	Examples: • `lrcmd -cmd=stop -jobid=MyApp:1 -port=80 -host=myodrhost.com` • `lrcmd -cmd=stop -jobid=MyApp:1 -port=9080 -host=mygshost.com -userid=myname -password=mypassword`

Table 9. lrcmd commands (continued). The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional information
Show the symbolic variables that are referenced in the job definition xJCL.	`-cmd=getSymbolicVariables -xJCL=<xjcl_file> [-<host>] [-port=<port>]`	This command shows the symbolic variables which are referenced in the job definition xJCL. Use `-jobid=<jobid>` as the job ID assigned to the job by the job scheduler Optional arguments: • Use `-XJCL=<xjcl_file>` to specify the path of the job definition xJCL file which describes the grid job. • Use `-job=<job_name>` to specify the job name, which is a key in the job repository of the job scheduler. • Use `-host=<host>` as the ODR host name or job scheduler server host name. If not specified, default is localhost. • Use `-port=<port>` as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.	Examples: • `lrcmd -cmd=getSymbolicVariables -xJCL=C:\myXJCL -port=9080 -host=mygshost.com` • `lrcmd -cmd=getSymbolicVariables -job=MyJob -port=80 -host=myodrhost.com -userid=myname -password=mypassword`
Save the job log.	`-cmd=saveJobLog -jobid=<job_id> [-host=<host>] [-fileName=<fileName>]`	This command saves the job log associated with the requested job identifier to the local file system. Use `-jobid=<job_id>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The Use `-fileName=<fileName>` to indicate the name of a file on the local file system where the compressed job log data is to be saved. The file is replaced if it exists. The file name `<fileName>`might not contain embedded blanks. Optional arguments: • Use `-host=<host>` as the ODR host name or job scheduler server host name. If not specified, the default is localhost. • Use `-port=<port>` as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.	Examples: • `lrcmd -cmd=saveJobLog -jobid=MyApp:1 -fileName=/tmp/myZippedJobLog -port=80 -host=myodrhost.com` • `lrcmd -cmd=saveJobLog -jobid=MyApp:1 -fileName=/tmp/mySavedJobLog -port=9080 -host=mygshost.com -userid=myname -password=mypassword`
Get job log.	`-cmd=getJobLog -jobid=<job_id>`	Displays the job log associated with the requested job identifier. Use `-jobid=<job_id>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. Optional arguments: • Use `-host=<host>` as the ODR host name or job scheduler server host name. If not specified, the default is localhost. • Use `-port=<port>` as the ODR HTTP Proxy address or job scheduler server HTTP port. If not specified, the default is 80.	Examples: • `lrcmd -cmd=getJobLog -jobid=MyApp:1 -port=80 -host=myodrhost.com` • `lrcmd -cmd=getJobLog -jobid=MyApp:1 -port=9080 -host=mygshost.com -userid=myname -password=mypassword`

Table 9. lrcmd commands (continued). The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional information
Purge job log	`-cmd=getJobLog -jobid=<job_id> -logTimeStamp=<logTimeStamp>`	Removes the job log associated with the requested job identifier and log time stamp. A job log entry remains in, for example: /opt/IBM/WebSphere/AppServer/profiles/scheduler/joblogs/PostingsSampleEar_99/14022007_164535/part.0.log. The entry tracks the reason why the job log was removed. • Use `-jobid=<job_id>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. • Use -logTimeStamp=<logTimeStamp to indicate the time stamp, the subdirectory name, which identifies the job log to be removed. The time stamp is returned by -cmd=getLogMetaData. • Use `-userid=<user_id>` to specify the user ID required when the job scheduler server is running in secure mode. • Use `-password=<password>` to specify the password required when the job scheduler server is running in secure mode.	Examples: • `lrcmd -cmd=getLogMetaData -jobid=PostingsSampleEar:99 -port=80 -host=myodrhost.com -userid=myname -password=mypassword` • `lrcmd -cmd=purgeJobLog -jobid=PostingsSampleEar:99 -port=80 -logTimeStamp=14022007_164535 -host=myodrhost.com -userid=myname -password=mypassword`
Display the job log metadata for the requested job identifier.	`-cmd=getLogMetaData -jobid=<job_id>`	The job log metadata indicates the log time stamps associated with the requested job identifier. The metadata or time stamp identifies a unique instance of the job. Logs from multiple different jobs with the same job number can exist. Use `-jobid=<job_id>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job.	Examples: • `lrcmd -cmd=getLogMetaData -jobid=MyApp:1 -port=80 -host=myodrhost.com` • `lrcmd -cmd=getLogMetaData -jobid=MyApp:1 -port=9080 -host=mygshost.com`
Display the job log part list.	`-cmd=getLogPartList -jobid=<job_id> -logTimeStamp=<logTimeStamp>`	Displays the job log part list associated with the requested job identifier and log time stamp. Use the command getLogMetaData to return a timestamp to use with `-logTimeStamp=<timestamp>`. Use `-jobid=<job_id>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job.	Examples: • `lrcmd -cmd=getLogPartList -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=80 -host=myodrhost.com` • `lrcmd -cmd=getLogPartList -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=9080 -host=myodrhost.com -userid=myname -password=mypassword`
Display the job log part.	`-cmd=getLogPart -jobid=<job_id> -logTimeStamp=<logTimeStamp> -logPart=<logPart>`	Displays the job log part associated with the requested job identifier, log time stamp, and log part. Use `-jobid=<job_id>` as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. Use `-logTimeStamp=<logTimeStamp>`to indicate the time stamp (the subdirectory name), which identifies the job log whose part list information is to be returned. The time stamp is returned by -cmd=getLogMetaData. Use `-logPart=<logPart>` to indicate the portion of the job log associated with the requested job identifier and time stamp to be returned. The log part information is returned by -cmd=getLogPartList.	Examples: • `lrcmd -cmd=submit -xJCL=myxjcl.xml -host=myhost -port=80` (returns a job identifier of PostingsSampleEar:99) • `lrcmd -cmd=getLogMetaData -jobid=PostingsSampleEar:99` (returns the timestamp 14022007_164535) • `lrcmd -cmd=getLogPart -jobid=PostingsSampleEar:99 -logTimeStamp=14022007_164535 -logPart=part.1.log`

Table 9. lrcmd commands (continued). The table includes arguments, a description, and additional information for the lrcmd command.

Command	Arguments	Description	Additional information
Display the size of the job log associated with the requested job identifier.	`-cmd=getLogSize -jobid=<job_id> -logTimeStamp=<logTimeStamp>`	This command returns the size of the job login bytes. Use -jobid=<job_id> as the job ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. Use `-logTimeStamp=<logTimeStamp>` to indicate the time stamp; that is, the subdirectory name, which identifies the job log whose part list information is to be returned. The time stamp is returned by -cmd=getLogMetaData.	Examples: • `lrcmd -cmd=getLogSize -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=80 -host=myodrhost.com` • `lrcmd -cmd=getLogSize -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=9080 -host=myodrhost.com -userid=myname -password=mypassword`
Return the age of the job log in the seconds since it was last modified.	`-cmd=getLogAge -jobid=<job_id> -logTimeStamp=<logTimeStamp>`	Displays the age of the • `lrcmd -cmd=getLogAge -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=80 -host=myodrhost.com` • `lrcmd -cmd=getLogAge -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=9080 -host=myodrhost.com -userid=myname -password=mypassword` Job log associated with the requested job identifier. Use `-jobid=<jobid>` as the ID assigned to the job by the job scheduler. The job ID is returned by the lrcmd -cmd=submit command that initially submitted the job. The -cmd=status command can also be used to identify the job ID for a particular job. Use `-logTimeStamp=<logTimeStamp>` to indicate the time stamp; that is, the subdirectory name, which identifies the job log whose part list information is to be returned. The time stamp is returned by -cmd=getLogMetaData.	Examples: • `lrcmd -cmd=getLogAge -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=80 -host=myodrhost.com` • `lrcmd -cmd=getLogAge -jobid=MyApp:1 -logTimeStamp=20102006_155529 -port=9080 -host=myodrhost.com -userid=myname -password=mypassword`

Example of retrieving output of a batch job:

```
lrcmd -cmd=output -jobid=mybatchjob:63 -host=myLRSHost -port=9081
```

```
CWLRB4940I: com.ibm.websphere.batch.wsbatch : -cmd=output -jobid=mybatchjob:63

CWLRB5000I: Wed Jun 15 17:55:36 EDT 2005 : com.ibm.websphere.batch.wsbatch : response to output

CWLRB1740I: [Wed Jun 15 17:55:36 EDT 2005] Job [mybatchjob:63] is in job setup.
CWLRB1760I: [Wed Jun 15 17:55:37 EDT 2005] Job [mybatchjob:63] is submitted for execution.
CWLRB2420I: [Wed Jun 15 17:55:37 EDT 2005] Job [mybatchjob:63] Step [Step1] is in step setup.
CWLRB2440I: [Wed Jun 15 17:55:38 EDT 2005] Job [mybatchjob:63] Step [Step1] is dispatched.
CWLRB2460I: [Wed Jun 15 17:55:38 EDT 2005] Job [mybatchjob:63] Step [Step1] is in step breakdown.
CWLRB2600I: [Wed Jun 15 17:55:38 EDT 2005] Job [mybatchjob:63] Step [Step1] completed normally rc=0.
CWLRB2420I: [Wed Jun 15 17:55:39 EDT 2005] Job [mybatchjob:63] Step [Step2] is in step setup.
CWLRB2440I: [Wed Jun 15 17:55:39 EDT 2005] Job [mybatchjob:63] Step [Step2] is dispatched.
CWLRB2460I: [Wed Jun 15 17:55:40 EDT 2005] Job [mybatchjob:63] Step [Step2] is in step breakdown.
CWLRB2600I: [Wed Jun 15 17:55:40 EDT 2005] Job [mybatchjob:63] Step [Step2] completed normally rc=4.
End
```

Job logs

A job log is a file that contains a detailed record of the execution details of a job. System messages from the batch container and output from the job executables are collected. By examining job logs, you can see the life cycle of a batch job, including output from the batch applications themselves.

A job log is composed of the following three types of information:

1. xJCL - A job log contains a copy of the xJCL used to run the job, including xJCL substitution values.

2. System messages - A set of system messages that communicate the major life cycle events corresponding to the job. The following system events are recorded in a job log:
   - Begin and end of a job
   - Begin and end of a step
   - Begin and end of a checkpoint
   - Open, close, and checkpoint of a batch data stream
   - Checkpoint algorithm invocation / results
   - Results algorithm invocation / results
3. Application messages - A set of messages written to standard out and standard error by a job step program.

Job logs are viewable through the job management console. Since information is added dynamically to the job log while the job is running, you can view the latest information by selecting Refresh from the job log view. Jobs logs are viewable only if the owning scheduler is active. In addition, if the endpoint running the job is unavailable, a partial job log is the result.

Output of a job log

Job log output is collected on the job scheduler node, and on the grid execution endpoint node. The output is collected in a directory which has the format:

`${GRID_JOBLOG_ROOT}/joblogs/<job-directory>/<timeStamp-directory>`

where

`${GRID_JOBLOG_ROOT}/joblogs` - The base directory for all job logs on the node. It is configurable through the endpoint job log location attribute of the job scheduler panel from the administration console. The default value for `${GRID_JOBLOG_ROOT}` is `${user.install.root}`.

`<job-directory>` Is generated at run time from the job name. For example, if the job ID assigned by the job scheduler is `PostingsSampleEar:99`, then the generated directory name is `PostingsSampleEar_99`

`<timeStamp-directory>`- Is generated at run time from the current date. It is in the format ddmmyyy_hhmmss, where dd is the day of the month, mm is a month (00 - 11), and yyyy is the year. hh is the hour of the day (00 - 23), mm is the minute of the hour (00 - 59) and ss is the seconds of the minutes (00 - 59). For example, a timestamp directory with the name 14022007_164535 means that the job began processing on 14 Mar 2007, at 16:45:35.

For example, job output from job PostingsSampleEar:99 might be collected in the directory `/opt/IBM/WebSphere/AppServer/profiles/scheduler/joblogs/PostingsSampleEar_99/14022007_164535`.

Output on the scheduler node contains an echo of the job xJCL (before and after symbolic variable substitution, if any, is performed) and job dispatch information. Job log output from the job scheduler is collected in the job log directory in the file named part.0.log. Output on the execution endpoint node contains both application output and grid endpoint runtime messages. This output includes any application generated output directed to the System.out and System.err output streams. Job log output from the grid endpoint is collected in the job log directory in files with names such as part.1.log and part.2.log. However, if the job scheduler and grid endpoint are installed on the same application server, job log output from both the scheduler and the grid endpoint is collected in the job log directory in the file named part.0.log. Each of the log parts contains approximately 1000 records. The following example shows the contents of part.1.log:

```
System.out: [03/13/07 08:25:32:708 EDT] Tue Mar 13 08:25:32 EDT 2007: SimpleCI application starting...
System.out: [03/13/07 08:25:32:708 EDT] -->Will loop processing a variety of math functions for approximately 30.0 seconds!
System.out: [03/13/07 08:26:02:752 EDT] Tue Mar 13 08:26:02 EDT 2007: SimpleCI application complete!
System.out: [03/13/07 08:26:02:753 EDT] -->Actual Processing time = 30.043 seconds!
CWLRB5764I: [03/13/07 08:26:03:069 EDT] Job SimpleCIEar:44 ended
```

Administrative roles and privileges

Roles and privileges vary depending on your administrative role and the component.

Administrative roles and privileges

For definitions of administrative roles in WebSphere Application Server and how to assign them, see Authorizing access to administrative roles.

Table 10. Administrative roles and privileges. The table lists each component for the graphical user interface (GUI) and what privileges the component has for the monitor, operator, configurator, and administrator privileges.

GUI	Monitor privileges	Operator privileges	Configurator privileges	Administrator privileges
Job scheduler	View the information.	View the information.	Has all privileges.	Has all privileges.

Administrative console help

This reference information describes settings that you can view and configure on the pages of the product administrative console and elsewhere. Custom properties are documented separately. Custom properties are name-value pairs that you can enter on specific console pages if you know what to specify.

To open the information center table of contents to the location of this reference information, click **Show in Table of Contents** ( ) on your information center border. For example, if you found this page from the information center search or from an internet search engine, such as Google, click **Show in Table of Contents** ( ) to make the information center show you the location of this page in the information center navigation tree. You can then browse the contents that are indented under this navigation entry.

Custom property collection for the job scheduler

Use this page to view and manage arbitrary name-value pairs of data, where the name is a property key and the value is a string value that is used to set internal system configuration properties.

The administrative console contains several custom properties pages that work similarly. To view one of these administrative pages, click a custom properties link.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property that has that name is used. Do not start your property names with *was.* because this prefix is reserved for properties that are predefined in WebSphere Application Server.

Value:

Specifies the value that is paired with the specified name.

Description:

Provides information about the name and value pair.

Custom property settings for the job scheduler

Use this page to configure arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. You can define a new property to configure a setting beyond what is available in the administrative console.

To view this administrative console page, click **System administration** > **Job scheduler** > **Custom Properties**.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property that has that name is used. Do not start your property names with *was.* because this prefix is reserved for properties that are predefined in WebSphere Application Server.

Data type String

Value:

Specifies the value that is paired with the specified name.

Data type String

Description:

Provides information about the name and value pair.

Data type String

Job scheduler configuration

Use this page to set up persistence of job information to the external job database. Configuration settings include the deployment target of the job scheduler, the data source, the database schema name, and the endpoint job log location.

To view this administrative console page, click **System administration** > **Job scheduler**.

Use this page to add custom properties, map security roles to users and groups for the job scheduler, and view the grid endpoints.

Users in the lrsubmitter role can submit and manage their own jobs. Users in the lradmin role can perform any action against any job.

Scheduler hosted by:

Specifies the deployment target where the grid scheduler is hosted / running.

Database schema name:

Specifics the database schema name for the grid scheduler database.

Data source JNDI name:

Specifies the data source Java Naming and Directory Interface (JNDI) where grid jobs are stored.

Endpoint job log location:

Specifies a location on the endpoints where the job log is created.

WebSphere grid endpoints

View servers or clusters that host grid applications and the data source Java Naming and Directory Interface (JNDI) names used by the WebSphere grid endpoints.

To view this administrative console page, click **System administration** > **Job Scheduler** > **WebSphere grid endpoints**.

Name:

Specifies the name of the WebSphere grid endpoint.

Datasource JNDI name:

Specifies the data source JNDI name used by the WebSphere grid endpoint.

Welcome to the job management console

The job management console is a stand-alone web interface for users to perform job operations.

To view this job management console page, click **Help** from the Welcome page of the job management console.

You must be granted the lrsubmitter role, the lradmin role, or both roles through the administrative console to access the job management console.

Depending on role privileges, users can perform various job operations. The job management console provides controlled access when security is enabled.

View jobs

Use the page to view the list of all jobs submitted to the job scheduler and to view information about the jobs.

To view this job management console page, click **Job Management** > **View jobs**.

To apply action on the jobs, select the jobs and the action, then click **Apply**.

Use the filter function to filter and sort jobs. Click the Show filter function icon , enter the filter criteria, then click **Go**.

To refresh this panel, click **View Jobs** in the navigation or click the icon in the Status column.

Privileges in the job management console vary, depending on assigned roles. For this console page, a user can do the following actions:

- Manage jobs that the user owns with the lrsubmitter role.
- Manage jobs from all users with the lradmin role.

There are multiple types of actions to take, depending on the job type and state. From this console page, users with the lrsubmitter or lradmin roles can complete actions such as cancel, purge, restart, remove, suspend, or resume a job.

Job ID:

Specifies the link to the job. To view the job log, click the link of the job ID.

Submitter:

Specifics the ID of the user who submitted the job.

Last update:

Specifies the date when the job was last updated.

State:

Shows the state of the job.

Node:

Specifies the node where the job is located.

Application server:

Specifies the application server where the job is located.

View job log

Use this page to view job log information of all jobs submitted to the job scheduler, and to download a job log to your local system.

To view this job management console page, click **Job Management** > **View jobs** > ***job name***.

Use this panel to view job log information, or to download the log file to your local system.

Privileges in the job management console vary, depending on assigned roles. For this console page, a user can do the following actions:

- Manage jobs that the user owns with the lrsubmitter role.
- Manage jobs from all users with the lradmin role.

Refresh:

Click to refresh the screen.

Download:

Click to download this job log to your local system.

Back:

Click to return to the view job panel.

Submit a job

Use this page to submit a job by specifying the job definition. The job definition can originate from the local file system.

To view this job management console page, click **Job Management** > **Submit a job**.

Privileges in the job management console vary, depending on assigned roles. For this console page, users with the lrsubmitter role or the lradmin role can submit jobs.

Local file system:

Specify the path of the job definition to submit as a new job.

Use the **Browse** button to locate and specify the full path of a local file.

Substitution properties:

Select to update the values of the substitution properties for the job.

If a job has substitution properties without values, you must specify them.

Custom properties at different scopes

Custom properties are unique settings in the administrative console. They are name-value pairs that you can enter on specific administrative console pages. Use the documentation for the specific custom property to determine where the property must be set.

Setting cell-wide custom properties

You can configure a cell-wide custom property using the following steps:

1. In the administrative console, select **System administration** > **Cell** > **Custom properties** > **New**.
2. Provide a name and value for the custom property.
3. Click **OK**.

Setting custom properties at other levels

Some custom properties can be set at other levels than the cell level. If the custom property documentation specifies another scope at which the custom property must be set, the steps are similar, but you first go to the indicated level before you set the custom property.

1. Go to the indicated configuration object.
2. Click **Custom properties** > **New**.
3. Provide a name and value for the custom property.
4. Click **OK**.

Job scheduler custom properties

Custom properties modify the job scheduler configuration. You can use these settings to tune the job scheduler behavior beyond the settings that are in the administrative console.

You can use the custom properties page to define the following job scheduler custom properties:

- "MaxConcurrentDispatchers"
- "UseHTTPSConnection"
- "RECORD_SMF_SUBTYPES" on page 38

MaxConcurrentDispatchers:

Define this custom property if jobs are being dispatched slowly when large numbers of jobs are submitted. By default, MaxConcurrentDispatchers is set to 100. MaxConcurrentDispatchers is an optional custom property.

Table 11. MaxConcurrentDispatchers custom property values. The table includes the scope, valid values, and default for the custom property.

Scope	Valid values	Default
Job scheduler	Integer value greater than 0	100

UseHTTPSConnection:

Define this custom property if you want to enable HTTP SSL connections between the job scheduler and the common batch container. By default, HTTP SSL connections are disabled. UseHTTPSConnection is an optional custom property.

Table 12. UseHTTPSConnection custom property values. The table includes the scope, valid values, and default for the custom property.

Scope	Valid values	Default
Job scheduler	• true Enables HTTP SSL connections • false Disables HTTP SSL connections	false (disabled)

RECORD_SMF_SUBTYPES:

Define this property to indicate which SMF 120 record subtype you want to use to record job usage data. By default, SMF 120 subtype 20 records are used. RECORD_SMF_SUBTYPES is an optional custom property.

Table 13. RECORD_SMF_SUBTYPES custom property values. The table includes the scope, valid values, and default for the custom property.

Scope	Valid values	Default
Job scheduler cell	• 20 Use SMF120 subtype 20 records. • 9 Use SMF120 subtype 9 records. **Note:** SMF120 subtype 9 support for batch jobs is available on WebSphere® Application Server Version 8 or later. Earlier versions are not supported. If you specify 9 on an earlier version, the job scheduler issues a message. The message notifies you that SMF120 subtype 9 is not supported on the earlier version of WebSphere Application Server. The job scheduler reverts to SMF120 subtype 20 records. **Note:** SMF 120 subtype 9 support for batch jobs requires that SMF 120 subtype 9 recording for asynchronous beans is enabled on the endpoint server. SMF 120 subtype 9 support for asynchronous beans is available on WebSphere Application Server Version 8.0.0.1 or later. Earlier versions are not supported. If you specify `RECORD_SMF_SUBTYPES=9` on an earlier version, the job scheduler issues a message. The message indicates that SMF 120 subtype 9 records are not supported on earlier versions of WebSphere Application Server. The job scheduler reverts to SMF 120 subtype 20 records. **Note:** If you specify `RECORD_SMF_SUBTYPES=9` without also enabling SMF 120 subtype 9 recording for asynchronous beans in the endpoint server, the endpoint server issues a message. The message indicates that SMF 120 subtype 9 recording for asynchronous beans is not enabled. No SMF120 subtype 9 job usage records are collected. • ALL Use both SMF120 subtype 20 and SMF120 subtype 9 records.	20

Batch administrator examples

Batch administrator examples are examples of code snippets, command syntax, and configuration values that are relevant to performing administrative and deployment tasks in the batch environment.

The samples in this section consist of xJCL samples and XML schemas for batch jobs, native jobs, and compute-intensive jobs.

xJCL sample for a batch job

The following sample illustrates a batch job, which demonstrates that you can invoke existing session beans from within job steps.

```
<job name="PostingsSampleEar" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

        <jndi-name>ejb/com/ibm/websphere/samples/PostingsJob</jndi-name>


    <step-scheduling-criteria>
  <scheduling-mode>sequential</scheduling-mode>
 </step-scheduling-criteria>


    <checkpoint-algorithm name="${checkpoint}">
  <classname>com.ibm.wsspi.batch.checkpointalgorithms.${checkpoint}</classname>
  <props>
   <prop name="interval" value="${checkpointInterval}" />
  </props>
 </checkpoint-algorithm>


    <results-algorithms>
  <results-algorithm name="jobsum">
   <classname>com.ibm.wsspi.batch.resultsalgorithms.jobsum</classname>
  </results-algorithm>
 </results-algorithms>


    <substitution-props>
        <prop name="wsbatch.count" value="5" />
        <prop name="checkpoint" value="timebased" />
        <prop name="checkpointInterval" value="15" />
        <prop name="postingsDataStream" value="${was.install.root}${file.separator}temp${file.separator}postings" />
    </substitution-props>

    <job-step name="Step1">


        <jndi-name>ejb/DataCreationBean</jndi-name>

        <!-- apply checkpoint policy to step1 -->
        <checkpoint-algorithm-ref name="${checkpoint}" />


        <results-ref name="jobsum"/>


  <batch-data-streams>
   <bds>

                <logical-name>myoutput</logical-name>


                <impl-class>com.ibm.websphere.samples.PostingOutputStream</impl-class>

                <props>
```

```
                  <prop name="FILENAME" value="${postingsDataStream}" />

   </props>
  </bds>
 </batch-data-streams>


       <props>

           <prop name="wsbatch.count" value="${wsbatch.count}" />
 </props>
</job-step>


<job-step name="Step2">


       <step-scheduling condition="OR">
  <returncode-expression step="Step1" operator="eq" value="0" />
           <returncode-expression step="Step1" operator="eq" value="4" />
 </step-scheduling>

       <jndi-name>ejb/PostingAccountData</jndi-name>
 <checkpoint-algorithm-ref name="${checkpoint}" />
 <results-ref name="jobsum"/>

 <batch-data-streams>
  <bds>

              <logical-name>myinput</logical-name>
   <impl-class>com.ibm.websphere.samples.PostingStream</impl-class>


              <props>
    <prop name="FILENAME" value="${postingsDataStream}" />
   </props>

  </bds>
 </batch-data-streams>
</job-step>

        <job-step name="Step3">
      <step-scheduling>
          <returncode-expression step="Step2" operator="eq" value="4" />
      </step-scheduling>

      <jndi-name>ejb/OverdraftAccountPosting</jndi-name>
      <checkpoint-algorithm-ref name="${checkpoint}" />
      <results-ref name="jobsum" />


     <batch-data-streams>
          <bds>

             <logical-name>dbread</logical-name>
              <impl-class>com.ibm.websphere.samples.OverdraftInputStream</impl-class>
         </bds>
     </batch-data-streams>
  </job-step>
</job>
```

XML schema for a compute intensive job

The following example shows the XML schema for a compute-intensive job:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
```

```
<xsd:element name="classname" type="xsd:string" />
<xsd:element name="jndi-name" type="xsd:string" />

<xsd:element name="required-capability">
   <xsd:complexType>
       <xsd:attribute name="expression" type="xsd:string" use="required" />
   </xsd:complexType>
</xsd:element>

<xsd:element name="substitution-props">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element ref="prop" minOccurs="0" maxOccurs="unbounded"/>
       </xsd:sequence>
   </xsd:complexType>
</xsd:element>

<xsd:element name="job-scheduling-criteria">
  <xsd:complexType>
      <xsd:sequence>
          <xsd:element maxOccurs="unbounded" minOccurs="1" ref="required-capability" />
      </xsd:sequence>
  </xsd:complexType>
</xsd:element>

<xsd:element name="job">
   <xsd:complexType>
     <xsd:sequence>
         <xsd:element ref="jndi-name" minOccurs="1" maxOccurs="1"/>
         <xsd:element ref="job-scheduling-criteria" minOccurs="0" maxOccurs="1"/>
         <xsd:element ref="substitution-props" minOccurs="0" maxOccurs="1"/>
         <xsd:element ref="job-step" maxOccurs="unbounded" minOccurs="1" />
     </xsd:sequence>
     <xsd:attribute name="name" type="xsd:string" use="required" />
     <xsd:attribute name="class" type="xsd:string" use="optional" />
     <xsd:attribute name="accounting" type="xsd:string" use="optional" />
     <xsd:attribute name="default-application-name" type="xsd:string" use="optional" />
   </xsd:complexType>
</xsd:element>

<xsd:element name="job-step">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element ref="classname" minOccurs="1" maxOccurs="1"/>
           <xsd:element ref="props" minOccurs="0" maxOccurs="1"/>
       </xsd:sequence>
       <xsd:attribute name="name" type="xsd:string" use="optional" />
       <xsd:attribute name="application-name" type="xsd:string" use="optional" />
   </xsd:complexType>
</xsd:element>


<xsd:element name="prop">
   <xsd:complexType>
       <xsd:attribute name="name" type="xsd:string" use="required" />
       <xsd:attribute name="value" type="xsd:string" use="required" />
   </xsd:complexType>
</xsd:element>

<xsd:element name="props">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element ref="prop" maxOccurs="unbounded" minOccurs="0"/>
       </xsd:sequence>
```

```
   </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

XML schema for a batch job

The following example shows the XML schema for a batch job:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <xsd:element name="classname" type="xsd:string" />
 <xsd:element name="impl-class" type="xsd:string" />
 <xsd:element name="jndi-name" type="xsd:string" />
 <xsd:element name="logical-name" type="xsd:string" />

 <xsd:element name="scheduling-mode">
     <xsd:simpleType>
         <xsd:restriction base="xsd:string">
             <xsd:pattern value="sequential"/>
         </xsd:restriction>
     </xsd:simpleType>
 </xsd:element>

 <xsd:element name="required" >
     <xsd:simpleType>
         <xsd:restriction base="xsd:string">
             <xsd:pattern value="[YNyn]"/>
         </xsd:restriction>
     </xsd:simpleType>
 </xsd:element>

 <xsd:element name="batch-data-streams">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element maxOccurs="unbounded" minOccurs="1" ref="bds" />
       </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

 <xsd:element name="job-scheduling-criteria">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element maxOccurs="unbounded" minOccurs="1" ref="required-capability" />
       </xsd:sequence>
   </xsd:complexType>
 </xsd:element>

 <xsd:element name="bds">
    <xsd:complexType>
        <xsd:all>
            <xsd:element ref="logical-name" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="impl-class" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="props" minOccurs="0" maxOccurs="1"/>
        </xsd:all>
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="checkpoint-algorithm">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="classname" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="props" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required" />
    </xsd:complexType>
 </xsd:element>
```

```
<xsd:element name="checkpoint-algorithm-ref">
   <xsd:complexType>
       <xsd:attribute name="name" type="xsd:string" use="required" />
   </xsd:complexType>
</xsd:element>

<xsd:element name="required-capability">
   <xsd:complexType>
       <xsd:attribute name="expression" type="xsd:string" use="required" />
   </xsd:complexType>
</xsd:element>

<xsd:element name="results-algorithm">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element ref="classname" minOccurs="1" maxOccurs="1"/>
           <xsd:element ref="props" minOccurs="0" maxOccurs="1"/>
           <xsd:element ref="required" minOccurs="0" maxOccurs="1"/>
       </xsd:sequence>
       <xsd:attribute name="name" type="xsd:string" use="required" />
   </xsd:complexType>
</xsd:element>

<xsd:element name="results-algorithms">
       <xsd:complexType>
           <xsd:sequence>
               <xsd:element maxOccurs="unbounded" minOccurs="1" ref="results-algorithm" />
           </xsd:sequence>
       </xsd:complexType>
</xsd:element>

<xsd:element name="results-ref">
       <xsd:complexType>
           <xsd:attribute name="name" type="xsd:string" use="required" />
       </xsd:complexType>
</xsd:element>

<xsd:element name="substitution-props">
   <xsd:complexType>
       <xsd:sequence>
           <xsd:element ref="prop" minOccurs="0" maxOccurs="unbounded"/>
       </xsd:sequence>
   </xsd:complexType>
</xsd:element>


<xsd:element name="job">
   <xsd:complexType>
     <xsd:sequence>
         <xsd:element ref="jndi-name" minOccurs="1" maxOccurs="1"/>
         <xsd:element ref="job-scheduling-criteria" minOccurs="0" maxOccurs="1"/>
         <xsd:element ref="step-scheduling-criteria" minOccurs="0" maxOccurs="1"/>
         <xsd:element ref="checkpoint-algorithm" maxOccurs="unbounded"  minOccurs="1"/>
         <xsd:element ref="results-algorithms" maxOccurs="1"  minOccurs="0"/>
         <xsd:element ref="substitution-props" minOccurs="0" maxOccurs="1"/>
         <xsd:element ref="job-step" maxOccurs="unbounded" minOccurs="1" />
     </xsd:sequence>
     <xsd:attribute name="name" type="xsd:string" use="required" />
     <xsd:attribute name="class" type="xsd:string" use="optional" />
     <xsd:attribute name="accounting" type="xsd:string" use="optional" />
     <xsd:attribute name="default-application-name" type="xsd:string" use="optional" />
   </xsd:complexType>
</xsd:element>

<xsd:element name="job-step">
   <xsd:complexType>
```

```
        <xsd:sequence>
            <xsd:element ref="step-scheduling" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="jndi-name" minOccurs="1" maxOccurs="1"/>
            <xsd:element ref="checkpoint-algorithm-ref" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="results-ref" minOccurs="0" maxOccurs="unbounded"/>
            <xsd:element ref="batch-data-streams" minOccurs="0" maxOccurs="1"/>
            <xsd:element ref="props" minOccurs="0" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="optional" />
        <xsd:attribute name="application-name" type="xsd:string" use="optional" />
    </xsd:complexType>
 </xsd:element>


 <xsd:element name="prop">
    <xsd:complexType>
        <xsd:attribute name="name" type="xsd:string" use="required" />
        <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="props">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="prop" maxOccurs="unbounded" minOccurs="0"/>
        </xsd:sequence>
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="returncode-expression">
    <xsd:complexType>
        <xsd:attribute name="step" type="xsd:string" use="required" />
        <xsd:attribute name="operator" type="xsd:string" use="required" />
        <xsd:attribute name="value" type="xsd:string" use="required" />
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="step-scheduling">
    <xsd:complexType>
         <xsd:sequence>
             <xsd:element ref="returncode-expression" minOccurs="1" maxOccurs="unbounded"/>
         </xsd:sequence>
         <xsd:attribute name="condition" type="xsd:string" use="optional" />
    </xsd:complexType>
 </xsd:element>

 <xsd:element name="step-scheduling-criteria">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="scheduling-mode" minOccurs="1" maxOccurs="1" />
        </xsd:sequence>
    </xsd:complexType>
 </xsd:element>

</xsd:schema>
```

Chapter 5. Administering Client applications

This page provides a starting point for finding information about application clients and client applications. Application clients provide a framework on which application code runs, so that your client applications can access information on the application server.

For example, an insurance company can use application clients to help offload work on the server and to perform specific tasks. Suppose an insurance agent wants to access and compile daily reports. The reports are based on insurance rates that are located on the server. The agent can use application clients to access the application server where the insurance rates are located. More introduction...

Deploying client applications

Deploying a client application depends on installing appropriate supporting files on the client machine, usually some configuring actions, and adding the program files for the client application. When the client application has been deployed, the application can run.

About this task

The steps required to deploy and run a client application depend on the type of client and the programming model used.

You can install an application client JAR file using the administrative console, wsadmin AdminApp install, or update commands. Install the client module only on a Version 8.0 deployment target (such as server, cluster, and so on).

Complete one or more of the following tasks:

Procedure

- Deploy the client application
- Run an ActiveX client application
- Deploy and run a Java EE client application
- Run the IBM Thin Client for Enterprise JavaBeans

Deploying applet client code

Applet clients are capable of communicating over the HTTP protocol and the RMI-IIOP protocol.

Before you begin

Applet clients have the following setup requirements:

- These clients are available on the Windows platforms. Check the prerequisites page for information on platform support and product prerequisites.
- The browser installation precedes the client code installation.

About this task

Unlike typical applets that are on web servers or WebSphere Application Servers and can only communicate over the HTTP protocol, applet clients can communicate over the HTTP protocol and the RMI-IIOP protocol. This additional capability gives the applet direct access to enterprise beans.

The applet container is the web browser and the Java plug-in combination. You must first install the Application Client for WebSphere Application Server so that the browser recognizes the IBM product Java plug-in.

Procedure

1. Install the Application Client for WebSphere Application Server.
2. Configure required Java runtime parameters.
   a. Click **Start** > **Control panel** .
   b. Select the IBM Control Panel for Java
   c. On the Advanced tab, enter the following parameter values in the Java Runtime Parameters field.

```
-Xmx512M
-Djava.security.policy=<app_client_root>\properties\client.policy
-Dwas.install.root=<app_client_root>
-Djava.ext.dirs=<app_client_root>\java\jre\lib\ext;
<app_client_root>\lib;
<app_client_root>\plugins;
<app_client_root>\lib\ext;
<app_client_root>\installedConnectors\
-Djava.class.path=<app_client_root>\properties
-Dcom.ibm.CORBA.ConfigURL=file:<app_client_root>\properties\sas.client.props
-Dcom.ibm.SSL.ConfigURL=file:<app_client_root>\properties\ssl.client.props
```

   Note: These parameter entries are automatically placed into the WebSphere Application Server control panel for the Java plug-in user who installed the WebSphere Application Server Application Client provided you are using a Java SE Development Kit (JDK) prior to JDK 1.5. If the applet is being run by a user other than the person who installed the client, then that user must enter the parameter entries.

   For JDK 1.5 and later, this automatic parameter feature is removed.

   - The Java Runtime Parameters field is similar to the command prompt when using command line options. Therefore, you can enter most options available from the command prompt (for example, -cp, classpath, and others) in this field as well.

3. Configure use of secure sockets layer (SSL) for secure access to resources. By default, the applet client is configured to have security enabled. If you have administrative security turned on at the server from which you are accessing resources, then you can use SSL when needed. If you decide that the security requirements for applet client applications differ from other types of client applications, then you can create special copies of client property files for applets to use.

Running an ActiveX client application

To run an ActiveX client application that is to use the ActiveX to Enterprise Java Beans (EJB) bridge, you must perform some initial configuration to set appropriate environment variables and to enable the ActiveX to EJB bridge to find its `XJB.JAR` file and the Java run time. This initial configuration sets up the environment within which the ActiveX client application can run.

About this task

To perform the required configuration, complete one or more of the following tasks:

Procedure

1. Start an ActiveX application and configure service programs.
2. Start an ActiveX application and configuring non-service programs

Starting an ActiveX application and configuring service programs

To run an ActiveX service program such as Active Server Page (ASP) that is to use the ActiveX to the Enterprise Java Bean (EJB) bridge, some initial configuration (to set appropriate environment variables and to enable the ActiveX to EJB bridge to find its `XJB.JAR` file and the Java run time) is necessary. This configuration sets up the environment within which the ActiveX service program can run.

Before you begin

The XJB.JClassFactory must find the Java run time dynamic link library (DLL) when initializing. In a service program such as Internet Information Server you cannot specify a path for its processes independently; you must set the process paths in the system PATH variable. This limitation means that you can only have a single Java virtual machine (JVM) version available on a machine using ASP.

About this task

To add the Java Runtime Environment (JRE) directories to your system path, complete one of the following task.

Procedure

On Windows XP systems, complete the following steps:
1. Open the Control Panel, then double-click the **System** icon.
2. Click the **Advanced** tab on the System Properties window.
3. Click **Environment Variables**.
4. Edit the Path variable in the System Variables window.
5. Add the following information to the beginning of the path that is displayed in the Variable Value field:

   `C:\WebSphere\AppClient\Java\jre\bin;C:\WebSphere\AppClient\Java\jre\bin\classic;`

   where `C:\WebSphere\AppClient` is the directory in which you installed the Java client in the WebSphere product.
6. Click **OK** in the Edit System Variable window to apply the changes.
7. Click **OK** in the Environment Variables window.
8. Click **OK** in the System Properties window.
9. Restart Windows XP.

What to do next

After you change the system PATH variable you must reboot the Internet Information Server machine so that Internet Information Server can see the change.

Starting an ActiveX application and configuring non-service programs

To run an ActiveX program initiated from an icon or command line (a non-service program) that is to use the ActiveX to the Enterprise Java Beans (EJB) bridge, you must perform some initial configuration to set appropriate environment variables and to enable the ActiveX to EJB bridge to find its `XJB.JAR` file and the Java run-time environment. This uses a batch file to set up the environment within which the ActiveX program can run.

About this task

To perform the required configuration, complete the following steps:

Procedure

1. Edit the `setupCmdLineXJB.bat` file to specify appropriate values for the environment variables required by the ActiveX to EJB bridge. For more information about these environment variables, see ActiveX to EJB bridge, environment and configuration. For more information about creating a JVM for an ActiveX program, see ActiveX to EJB bridge, initializing the Java virtual machine (JVM). After the ActiveX program has created an XJB.JClassFactory object and called the XJBInit() method, the JVM is initialized and ready for use.
2. Start the ActiveX client application by using one of the following methods:
   - Use the `launchClientXJB.bat` file to start the application. For example:

     `launchClientXJB MyApplication.exe` *parm1 parm2*

or

```
launchClientXJB MyApplication.vbp
```

- Use the `setupCmdLineXJB.bat` file to create an environment in which to run the application, then start the application from within that environment.

setupCmdLineXJB.bat, launchClientXJB.bat and other ActiveX batch files

This topic provides reference information about the aids that client applications and client services can use to access the ActiveX to EJB bridge. These enable the ActiveX to Enterprise JavaBeans (EJB) bridge to find its `XJB.JAR` file and the Java run-time environment.

Location

The include file is located in the *was_client_home*`\aspIncludes` directory. You can include the file into your Active Server Pages (ASP) application with the following syntax in your ASP page:

```
<--  #include virtual ="/WSASPIncludes/setupASPXJB.inc" -->
```

This syntax assumes that you have created a virtual directory in Internet Information Server called `WSASPIncludes` that points to the *was_client_home*`\aspIncludes` directory.

Usage notes

The following batch files are provided for client applications to use the ActiveX to EJB bridge:

- **setupCmdLineXJB.bat**

  Sets the client environment variables.
- **launchClientXJB.bat**

  Calls the `setupCmdLineXJB.bat` file and launches the application you specify as its arguments; for example:

  ```
  launchClientXJB.bat myapp.exe parm1 parm2
  ```

  or

  ```
  launchClientXJB MyApplication.vbp
  ```
- **Active Server Pages (ASP) include file**

  An include file is provided for ASP users to automatically set the following page-level (local) environment variables:
  - **com_ibm_websphere_javahome**. Path to the Java run-time directory installed with the WebSphere advanced server client.
  - **com_ibm_websphere_washome**. Path to the WebSphere advanced server client directory.
  - **com_ibm_websphere_namingfactory**. Sets the Java java.naming.factory.initial system property.
  - **com_ibm_websphere_computername**. (Optional) Name of the computer where the WebSphere Advanced Server Client is installed. If you intend to talk to a single specific computer, you are recommended to change this value to become the server name that you intend to access.
- **System settings**

  To enable the ActiveX to EJB bridge to access the Java run-time dynamic link library (DLL), the following directories must exist in the system PATH environment variable:

  was_client_home`\java\jre\bin;`*was_client_home*`\java\jre\bin\classic`

  Where *was_client_home* is the name of the directory where you installed the WebSphere Application Server client (for example, `C:\WebSphere\AppClient`).

  Note: This technique enables only one Java run time to activate on a machine, therefore all client services on that machine must use the same Java run time. Client applications do not have this limitation because they each have their own private, non-system scope.

Deploying and running a Java EE client application

You can use the launchClient command to run a Java Enterprise Edition (EE) client application in an Application Client installation or in a WebSphere Application Server node. Alternatively, you can use Java Web Start on a remote client machine to download and run a Java EE client application, including Thin client application, with a single click from a web browser on that machine.

Procedure

1. Deploy and run a Java EE client application for use with the launchClient command.

   After deploying a Java EE client application onto a machine with an Application Client installation or in a WebSphere Application Server node, you can start the application by using the launchClient command on that machine.

   a. Deploy the Java EE client application

   b. Start the Java EE client application

2. Deploy and run a Java EE client application by using Java Web Start.

   a. Prepare the Java EE client application ready to be deployed by remote action.

   b. Use Java Web Start on a remote client machine to download and run the Java EE client application.

Deploying a Java EE client application

Deploying a Java EE client application onto the client machines where it is to run includes distributing the EAR file for the client application and configuring resource references for use by the client application.

Before you begin

To run a deployed Java EE client application, the application needs access to a Application Client installation or a WebSphere Application Server installation.

For information about installing the Application Client on a client machine, refer to the Installing Application Client for WebSphere Application Server topic.

Attention: Application Client for WebSphere Application Server ships only with the 32-bit WebSphere Application Server.

About this task

Use this topic only if you later want to use the want to launchClient command to run the Java client application on an Application Client installation or in a WebSphere Application Server node.

If you want to download and run a Java EE client application remotely, you can use the Java Web Start to deploy the application onto the remote client machine with a single click from a Web browser on the client machine. For information about using Java Web Start to deploy Java EE client applications, see “Downloading and running a Java EE client application by using Java Web Start”.

Procedure

1. Distribute the EAR file.

   The client machines configured to run a client application must have access to the EAR file.

   - If all the machines in your environment share the same image and platform, run the Application Client Resource Configuration Tool (ACRCT) on one machine to configure the external resources, then distribute the configured EAR file to the other machines.
   - If your environment is set up with a variety of client installations and platforms, run the ACRCT for each unique configuration.
   - You can either distribute an EAR file to the correct client machines, or make it available on a network drive.

- Distributing EAR files is the responsibility of the system and network administrator.

2. Configure the resources for the application client. This generally involves using the Application Client Resource Configuration Tool (ACRCT) to configure references for the resources that the application is to use, including resource adapters, resource providers, data sources, and Java Message Service resources. These configurations are stored in the client JAR file within the application EAR file. The client runtime uses these configurations to resolve and create an instance of the resources for the client application.

   For some types of resources, other actions are needed; for example, to install a resource adapter and define environment variable needed to start the client application. More information about the actions for different types of resources is given in other configuring resources topics.

   If you plan to deploy the client application on z/OS, run the ACRCT on Windows. You can also run the ACRCT for distributed platforms locally.

   If the client application defines the local resources, but the resources are installed in a different location, run the ACRCT (clientConfig command) on the local machine to change the configuration in the EAR file. For example, the EAR file can contain a DB2 resource, configured as `C:\DB2`. If, however, you installed DB2 in the `D:\Program Files\DB2` directory, use the ACRCT to create a local version of the EAR file.

What to do next

After deploying the Java EE client application, use the launchClient command to run the client application.

Starting the Application Client Resource Configuration Tool and opening an EAR file:

You can perform many tasks by starting the Application Client Resource Configuration Tool (ACRCT). Many of these tasks also involve then opening an EAR file.

Before you begin

Attention: This task only applies to Java Platform, Enterprise Edition (Java EE) application clients.

About this task

Use these steps to start the Application Client Resource Configuration Tool. When you start the tool, one of the most common tasks that you perform is opening and modifying the components of EAR files.

Procedure

1. Open a command prompt and change to the *app_server_root*`\bin` directory.
2. Run the `clientConfig.bat` file.
3. Open an EAR file within the Application Client Resource Configuration Tool (ACRCT):
   a. Click **File** > **Open**.
   b. Select the file then click **Open**.
4. Save your changes to the file and close the tool:
   a. Click **File** > **Save**.
   b. Click **File** > **Exit**.

Deploying a resource adapter for a Java EE client application:

A Java EE client application can use a resource adapter to connect to an enterprise information system (EIS). To use a resource adapter, you need to install it, configure it, and configure related resources.

About this task

The resource adapter support provided for Java EE client applications is a subset of the support provided for application servers. A client resource adapter is used in a non-managed environment and must conform to the J2EE Connector Architecture Specification Version 1.5 or higher. Only outbound connections to the EIS are supported through the ManagedConnectionFactory interfaces. The inbound messaging support (from the EIS), life cycle management, and work management aspects of the specification are not supported on the client.

When running Java EE application clients, the launchClient script specifies a system property called **com.ibm.ws.client.installedConnector**, which is set to the same value as the *CLIENT_CONNECTOR_INSTALL_ROOT* variable. This is the default location for installed resource adapters and can be overridden for each launchClient call by specifying the **-CCD** parameter. When the client container is activated, all resource adapter subdirectories under the specified default location for the resource adapters directory are added to the classpath. This action allows the client application to use the resource adapters without using the ACRCT to specify any of the client resources.

Procedure

1. Install the resource adapter archive (RAR) file

   For a client application to use a resource adapter, the RAR file must be installed in the directory specified by the environment variable, CLIENT_CONNECTOR_INSTALL_ROOT, defined when the setupCmdLine script runs. The launchClient tool, Application Client Resource Configuration Tool (ACRCT) and clientRAR tool all use this variable to find the default location of all installed resource adapters.

   To install a RAR file for a client application, use the clientRAR tool.
2. Configure the resource adapter and its resources for the client application Use the Application Client Resource Configuration Tool (ACRCT) to define the resource adapter, connection factories, and administered objects in the EAR file for the client application. The client application uses this configuration to resolve and create an instance of the resource adapter and the other resources.
   a. Configure the resource adapter
   b. Configure a connection factory
   c. Configure administered objects

clientRAR tool:

This topic describes the command line syntax for the client resource adapter installation tool.

If this tool is used to add or delete resource adapters on the server, then only the client can use the resource adapter. If the resource adapter is installed on the server using the wsadmin tool or the administrative console, then do not use the clientRAR tool remove it. Only resource adapters that are installed using the clientRAR tool should be removed using the clientRAR tool.

The command line invocation syntax for the `clientRAR` tool follows:

```
clientRAR [-help | -?] [-CRDcom.ibm.ws.client.installedConnectors=<dir>] <task> <archive>

where
-help, -?
Print the usage information.
-CRDcom.ibm.ws.client.installedConnectors
The directory where resource adapters are installed.
This will override the system property of the same name
(com.ibm.ws.client.installedConnectors).


<task>
The task to perform: add - install, delete - uninstall.
```

```
<archive>
if task=add then this is the fully qualified name of the resource adapter archive file.
If task=delete then this is the filename of the resource adapter archive to be uninstalled.
```

The following examples demonstrate correct syntax.

Configuring resource adapters for the client:

Use the Application Client Resource Configuration Tool (ACRCT) to configure resource adapters for the client.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT).
2. Open the EAR file for which you want to configure new resource adapters. The EAR file contents display in a tree view.
3. Select the JAR file in which you want to configure the new resource adapters from the tree.
4. Expand the JAR file to view its contents.
5. Right-click the Resource Adapters folder, and click **New**.
6. Configure the resource adapter settings in the resulting property dialog.
7. Click **OK**.
8. Click **File > Save** on the menu bar to save your changes.

Resource adapters for the client:

A resource adapter is a system-level software driver that a Java application uses to connect to an enterprise information system (EIS). A resource adapter plugs into an application client and provides connectivity between the EIS and the enterprise application.

Important: This topic is not relevant to the WebSphere MQ resource adapter. WebSphere MQ classes are picked up automatically by the client container (for both stand alone and with a WebSphere Application Server installation).

The resource adapter support for the Java EE client applications is a subset of the support for the server. For any resource adapter installed using the clientRAR tool, the client resource adapter is used in a non-managed environment and must conform to the Java EE Connector Architecture Specification Version 1.5 or higher. Only outbound connections to the EIS are supported through the ManagedConnectionFactory interfaces. The inbound messaging support (from the EIS), life cycle management, and work management aspects of the specification are not supported on the client.

For a client application to use a resource adapter, it must be installed in the directory specified by the environment variable, CLIENT_CONNECTOR_INSTALL_ROOT, defined when the setupCmdLine script runs. The launchClient tool, Application Client Resource Configuration Tool (ACRCT) and clientRAR tool all use this variable to find the default location of all installed resource adapters. To install a resource adapter in the client, use the clientRAR tool. Once the resource adapter is installed, it must be configured using the ACRCT. The client configuration tool adds the resource adapter configuration to the EAR file. Then, connection factories and administered objects are defined.

When running Java EE application clients, the `launchClient` script specifies a system property called `com.ibm.ws.client.installedConnector`, which is set to the same value as the `CLIENT_CONNECTOR_INSTALL_ROOT` variable. This is the default location for installed resource adapters and can be overridden for each `launchClient` call by specifying the `-CCD` parameter. When the client container is activated, all resource adapter subdirectories under the specified default location for the resource adapters directory are added to the classpath. This action allows the client application to use the resource adapters without using the ACRCT to specify any of the client resources.

Using resource adapters is a new mechanism for easily extending client applications.

Resource adapter settings:

Use this panel to view or change the configuration properties of the resource adapter. These configuration properties control how resource adapters are created.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Resource Adapter**. Right-click **Resource Adapter** and click **New**. The following fields appear on the **General** tab.

Name:

The name by which this Resource Adapter is known for administrative purposes within IBM WebSphere Application Server. The name must be unique within the Resource Adapters across the product administrative domain.

Data type String

Description:

A description of this resource adapter for administrative purposes within IBM WebSphere Application Server.

Data type String

Class Path:

Any additional class path. The path to the resource adapter directory is automatically added.

Data type String
Default The path to your Resource Adapter directory.

Native Path:

The native path where the Resource Adapter is located. Enter any additional native class path here.

Data type String

Resource Adapter Name:

A mandatory field that points to an installed resource adapter subdirectory. The entry does not represent the full directory name for the resource adapter. The full directory name is the installed resource adapter path, plus the resource adapter name.

Data type String

Installed Resource Adapter Path:

The directory where resource adapters are installed. If you do not complete this field, then the default takes effect.

If you specify the value, `${CONNECTOR_INSTALL_ROOT}`, then this value replaces the value of the `CLIENT_CONNECTOR_INSTALL_ROOT` variable on the machine on which the client application runs. This action allows the application to run easily on different machines, where the client installation might be in different locations.

Data type String
Default ${CONNECTOR_INSTALL_ROOT}

Configuring new connection factories for resource adapters for the client:

Use the Application Client Resource Configuration Tool (ACRCT) to configure new connection factories for resource adapters for the client.

About this task

Complete this task to configure new connection factories for resource adapters.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT).
2. Open the EAR file for which you want to configure new connection factories. The EAR file contents display in a tree view.
3. Select the JAR file in which you want to configure the new connection factories from the tree.
4. Expand the JAR file to view its contents.
5. Click the Resource Adapters folder.
6. Expand the resource adapter for which you want to create connection factories.
7. Right-click the Connection Factories folder and click **New**.
8. Configure the connection factory properties in the resulting property dialog.
9. Click **OK**.
10. Click **File > Save** on the menu bar to save your changes.

Resource adapter connection factory settings:

Use this panel to view or change the configuration properties of the selected resource adapter connection factory.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Resource Adapters**. Right-click the **Connection Factories** folder, and click **New**. The following fields appear on the **General** tab.

Name:

The name by which this connection factory is known for administrative purposes within WebSphere Application Server. The name must be unique within the resource adapter connection factories across the product administrative domain.

Data type String

Description:

An optional description of this connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String

JNDI Name:

The JNDI name that is used to match this resource adapter connection factory definition to the deployment descriptor. This entry should be a `resource-ref` name.

Data type String

User Name:

The **User Name** used, with the **Password** property, for authentication if the calling application does not provide a `userid` and `password` explicitly when getting a connection. If this field is used, then the Properties field `UserName` is ignored.

If you specify a value for the **User Name** property, you must also specify a value for the **Password** property.

The connection factory **User Name** and **Password** properties are used if the calling application does not provide a `userid` and `password` explicitly when getting a connection.

Data type String

Password:

Specifies an encrypted password. If you complete this field, then the **Password** field in the Properties box is ignored.

If you specify a value for the **UserName** property, you must also specify a value for the **Password** property.

Data type String

Re-Enter Password:

Confirms the password.

Type:

A drop-down list of all the `connectionFactoryInterfaces` as defined for the factories in the Resource Adapter Archive.

For each **Type**, there is a set of properties specified in the Properties box. This set of properties is constructed by retrieving the properties from each connection definition object. For any existing connection factories that are displayed for updating, this list of properties is overlaid with the properties specified for the objects. When the **Type** field is changed, the properties also change to reflect the correct properties for that type.

Data type String

Configuring administered objects for resource adapters for the client:

This section helps you configure new administered objects for the client.

Before you begin

Before you configure new administered objects, you must complete the following prerequisites:

1. Install the Resource Adapter Archive file (RAR) using the clientRAR tool.
2. Configure the resource adapter for the `.ear` file, using the Application Client Resource Configuration Tool (ACRCT) tool.

About this task

Complete this task to configure new administered objects for installed resource adapters.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT).
2. Open the EAR file for which you want to configure new administered objects. The EAR file contents display in a tree view.
3. Select the JAR file in which you want to configure the new administered objects from the tree.
4. Expand the JAR file to view its contents.
5. Click the **Resource Adapters** folder.
6. Expand the resource adapter for which you want to create administered objects.
7. Right-click the **Administered Objects** folder and click **New**.
8. Configure the administered object properties in the resulting property dialog.
9. Click **OK**.
10. Click **File > Save** on the menu bar to save your changes.

Administered objects settings:

Use this panel to view or change the configuration properties of the selected administered objects.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Resource Adapters** > *resource_adapter_instance*. Right-click **Administered Objects** and click **New**. The following fields appear on the **General** tab.

The settings for administered objects are handled similarly to connection factories. When updating administered objects, use the same panels that you used to create administered objects.

Name:

The name by which this administered object is known for administrative purposes within IBM WebSphere Application Server. The name must be unique within the resource adapter administered objects across the product administrative domain.

Data type String

Description:

An optional description of this connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String

JNDI Name:

This entry is a `resource-env-ref` name, a `message-destination-ref` name (if the `message-destination-ref` has no link), or a `message-destination` link.

Data type String

Type:

A drop-down list of all the administered object class-interface pairs as defined for the admin objects in the Resource Adapter Archive (RAR) file.

For each **Type**, there is a set of properties specified in the Properties box. This set of properties is constructed by retrieving the properties from each administered object definition. For any existing administered objects that are displayed for updating, this list of properties is overlaid with the properties specified for the objects. When the **Type** field is changed, the properties also change to reflect the correct properties for that type.

Data type String

Enabling client use of data sources:

If a Java EE client application accesses a database directly, you must provide the database drivers on the client machine, and configure the data source provider (JDBC provider) and data sources. Instead of accessing the database directly, it is recommended that your client application access the database through an enterprise bean.

About this task

WebSphere Application Server and the Application Client for WebSphere Application Server do not provide client database drivers to be used directly from a Java EE client application. You can contact your database vendor to get client database driver code and licenses.

Data sources configured on the server and looked up on the client do not participate in global transactions.

Instead of accessing the database directly, it is recommended that your client application access the database through an enterprise bean. This technique eliminates the need to have database drivers on the client machine, because the database access is handled by the enterprise bean running on WebSphere Application Server. It also enables the client application to take advantage of the pooling and additional database functions provided by the server.

For a current list of data source providers that are supported on WebSphere Application Server, see the WebSphere Application Server prerequisite website.

Procedure

1. For direct access from a client to the database, install the client database drivers on the client machine. For information about installing database drivers, see the documentation provided by your database vendor.
2. Configure a data source provider and a data source for the client application Use the Application Client Resource Configuration Tool (ACRCT) to define the data source provider and a data source in the EAR file for the client application. The client application uses this configuration to resolve and create an instance of the data source provider and data source.
   a. Configure a new data source provider. This provider describes the JDBC database implementation for your client application.
   b. Configuring a new data source This describes the client properties of the database your client application uses.

Configuring new data source providers (JDBC providers) for application clients:

You can create new data source providers, also known as JDBC providers, for your application client using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

During this task, you create new data source providers, also known as JDBC providers, for your application client. In a separate administrative task, install the Java code for the required data source provider on the client machine on which the application client resides.

About this task

Use this task to connect application clients to relational databases.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT) and open the EAR file for which you want to configure the new data source provider. The EAR file contents display in a tree view.
2. Select the JAR file in which you want to configure the new data source provider from the tree.
3. Expand the JAR file to view its contents.
4. Click the Data Source Providers folder. Do one of the following:
   - Right-click the folder and click **New Provider**.
   - Click **Edit > New** on the menu bar.
5. Configure the data source provider properties in the resulting property dialog.
6. Click **OK** when you finish.
7. Click **File > Save** on the menu bar to save your changes.

Example

You can configure data source provider and data source settings.

- Configuring data source provider and data source settings

The following code examples illustrates how to use configure data source provider and data source settings:

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1" name="jdbcProvider:name"
description="jdbcProvider:description" implementationClassName="jdbcProvider:
ImplementationClass">
<classpath>jdbcProvider:classPath</classpath>
<factories xmi:type="resources.jdbc:WAS40DataSource" xmi:id="WAS40DataSource_1"
name="jdbcFactory:name" jndiName="jdbcFactory:jndiName"
description="jdbcFactory:description" databaseName="jdbcFactory:databasename">
<propertySet xmi:id="J2EEResourcePropertySet_13">
<resourceProperties xmi:id="J2EEResourceProperty_13" name="jdbcFactory:customName"
value="jdbcFactory:customValue"/>
<resourceProperties xmi:id="J2EEResourceProperty_14" name="user"
value="jdbcFactory:user"/>
<resourceProperties xmi:id="J2EEResourceProperty_15" name="password"
value="{xor}NTs9PBk+PCswLSZlMT4yOg==/>
</propertySet>
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_14">
<resourceProperties xmi:id="J2EEResourceProperty_16" name="jdbcProvider:customName"
value="jdbcProvider:customeValue"/>
</propertySet>
</resources.jdbc:JDBCProvider>
```

- Required fields:
  - Data Source Provider Properties page: name

  - Data Source Properties page: name, jndiName
- Special cases:
  - The user name and password fields have no equivalent XMI tags. You must specify these fields in the custom properties.
  - The password is encrypted when you use the Application Client Resource Configuration Tool (ACRCT). If you do not use the ACRCT the field cannot be encrypted.

Example: Configuring data source provider and data source settings:

You can configure data source provider and data source settings.

The purpose of this article is to help you to configure data source provider and data source settings.
- Required fields:
  - Data Source Provider Properties page: name
  - Data Source Properties page: name, jndiName
- Special cases:
  - The user name and password fields have no equivalent XMI tags. You must specify these fields in the custom properties.
  - The password is encrypted when you use the Application Client Resource Configuration Tool (ACRCT). If you do not use the ACRCT the field cannot be encrypted.
- Example:

```
<resources.jdbc:JDBCProvider xmi:id="JDBCProvider_1" name="jdbcProvider:name"
description="jdbcProvider:description" implementationClassName="jdbcProvider:
ImplementationClass">
<classpath>jdbcProvider:classPath</classpath>
<factories xmi:type="resources.jdbc:WAS40DataSource" xmi:id="WAS40DataSource_1"
name="jdbcFactory:name" jndiName="jdbcFactory:jndiName"
description="jdbcFactory:description" databaseName="jdbcFactory:databasename">
<propertySet xmi:id="J2EEResourcePropertySet_13">
<resourceProperties xmi:id="J2EEResourceProperty_13" name="jdbcFactory:customName"
value="jdbcFactory:customValue"/>
<resourceProperties xmi:id="J2EEResourceProperty_14" name="user"
value="jdbcFactory:user"/>
<resourceProperties xmi:id="J2EEResourceProperty_15" name="password"
value="{xor}NTs9PBk+PCswLSZlMT4yOg==">
</propertySet>
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_14">
<resourceProperties xmi:id="J2EEResourceProperty_16" name="jdbcProvider:customName"
value="jdbcProvider:customeValue"/>
</propertySet>
</resources.jdbc:JDBCProvider>
```

Data source provider settings for application clients:

Use this page to create a data source under a JDBC provider which provides the specific JDBC driver implementation class.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file. Right-click **Data Source Providers** > and click **New**. The following fields appear on the **General** tab:

Name:

Specifies the display name for the data source.

For example you can set this field to *Test Data Source*.

Data type String

Description:

Specifies a text description for the resource.

Data type	String

Class Path:

A list of paths or `.jar` file names which together form the location for the resource provider classes.

Implementation class:

Use this setting to perform database specific functions.

Data type	String
Default	Dependent on JDBC driver implementation class

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Configuring new data sources for application clients:

Learn how to create date sources for application clients.

About this task

During this task, you create new data sources for your application client.

Procedure

1. Click the data source provider for which you want to create a data source in the tree. Take one of the following actions as needed:
   - Configure a new data source provider.
   - Click an existing data source provider.
2. Expand the data source provider to view its Data Sources folder.
3. Click the data source folder. Take one of the following actions as needed:
   - Right click the data source folder and click **New Factory**.
   - Click **Edit > New** on the menu bar.
4. Configure the data source properties in the displayed fields.
5. Click **OK** when you finish.
6. Click **File > Save** on the menu bar to save your changes.

Data source properties for application clients:

Use this page to create or modify the data sources.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Data Source Providers** > *Data source provider instance*. Right-click **Data Sources** and click **New**. The following fields are displayed on the **General** tab:

Name:

Specifies the display name of this data source.

Data type String

Description:

Specifies a text description of the data source.

Data type String

JNDI Name:

The application client run time uses this field to retrieve configuration information.

Database Name:

The name of the database to which you want to connect.

User:

Use the user ID with the Password property, for authentication if the calling application does not provide a user ID and password explicitly.

If you specify a value for the User ID property, then you must also specify a value for the Password property. The connection factory User ID and Password properties are used if the calling application does not provide a user ID and password explicitly.

Password:

Use the password with the User ID property, for authentication if the calling application does not provide a user ID and password explicitly.

If you specify a value for the Password property, then you must also specify a value for the User ID property.

Re-Enter Password:

Confirms the password.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Configuring mail providers and sessions for application clients:

You can edit the configurations of mail sessions and providers for your application clients using the Application Client Resource Configuration Tool (ACRCT).

About this task

Use the Application Client Resource Configuration Tool (ACRCT) to edit the configurations of mail sessions and providers for your application clients to use.

Procedure

1. Start the ACRCT.
2. Open an EAR file.
3. Locate the mail objects in the tree that is displayed for the EAR file. For example, if your file contains mail sessions, expand **Resources** > ***application*.jar** > **Mail Providers** > ***java_mail_provider_instance*** > **Mail Sessions**.

   In this example, ***java_mail_provider_instance*** is a particular mail provider.

Results

The mail session instances are located in the JavaMail Sessions folder.

Example

You can configure mail provider and mail session settings.

- Configuring mail provider and mail session settings for application clients

The following code examples illustrates how to configure mail provider and mail session settings for application clients:

```
<resources.mail:MailProvider xmi:id="builtin_mailprovider" name="Built-in Mail Provider" description="The built-in mail provider">
  <factories xmi:type="resources.mail:MailSession"
             xmi:id="MailSession_1207766754834" name="MailSession"
             jndiName="mail/session" description="Sample mail session" category="Sample"
             mailTransportHost="smtp.coldmail.com" mailTransportUser="transportUser"
             mailTransportPassword="{xor}Lz4sLChvLTs="
             mailFrom="smith@coldmail.com" mailStoreHost="imap.coldmail.com" mailStoreUser="storeUser"
             mailStorePassword="{xor}Lz4sLChvLTs="
             debug="true" strict="true"
             mailTransportProtocol="builtin_smtp" mailStoreProtocol="builtin_imap">
    <propertySet xmi:id="J2EEResourcePropertySet_1207766778585">
      <resourceProperties xmi:id="J2EEResourceProperty_1207766778585" name="key" type="java.lang.String" value="value" required="false"/>
    </propertySet>
  </factories>
  <protocolProviders xmi:id="builtin_smtp" protocol="smtp" classname="com.sun.mail.smtp.SMTPTransport" type="TRANSPORT"/>
  <protocolProviders xmi:id="builtin_pop3" protocol="pop3" classname="com.sun.mail.pop3.POP3Store" type="STORE"/>
  <protocolProviders xmi:id="builtin_imap" protocol="imap" classname="com.sun.mail.imap.IMAPStore" type="STORE"/>
  <protocolProviders xmi:id="builtin_smtps" protocol="smtps" classname="com.sun.mail.smtp.SMTPSSLTransport" type="TRANSPORT"/>
  <protocolProviders xmi:id="builtin_pop3s" protocol="pop3s" classname="com.sun.mail.pop3.POP3SSLStore" type="STORE"/>
  <protocolProviders xmi:id="builtin_imaps" protocol="imaps" classname="com.sun.mail.imap.IMAPSSLStore" type="STORE"/>
</resources.mail:MailProvider>
```

- Required fields:
  - Mail Provider Properties page: name, and at least one protocol provider
  - Mail Session Properties page: name, jndiName, outgoing server and protocol, and/or incoming server and protocol
- Special cases:
  - If you use the ACRCT tool, the password field will be encrypted. You cannot encrypt the password field if you do not use the ACRCT tool.

Mail provider settings for application clients:

Use this page to implement the JavaMail API and create mail sessions.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file. Right-click **Mail Providers** > and click **New**. The following fields appear on the **General** tab:

Name:

The name of the JavaMail resource provider.

Description:

An optional description for the resource provider.

Class Path:

Specifies a list of paths or JAR file names which together form the location for the resource provider classes.

Protocol:

Specifies the name of the protocol.

Classname:

Specifies the name of the class implementing the protocol. Leave this field blank if you want to use the default implementation.

Type:

This menu contains the following two values: `TRANSPORT` or `STORE`.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Mail session settings for application clients:

Use this page to configure mail session properties.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Mail Providers** > *mail provider instance*. Right-click **Mail Sessions** and click **New**. The following fields appear on the **General** tab:

Name:

Represents the administrative name of the JavaMail session object.

Description:

Provides an optional description for your administrative records.

JNDI Name:

The application client run time uses this field to retrieve configuration information.

Mail Transport Host:

Specifies the server to connect to when sending mail.

Mail Transport Protocol:

Specifies the transport protocol to use when sending mail.

Mail Transport User:

Specifies the user ID to use when the mail transport host requires authentication.

Mail Transport Password:

Specifies the password to use when the mail transport host requires authentication.

Enable strict Internet address parsing:

Specifies whether the recipient addresses must be parsed strictly in compliance with RFC 822, which is a specifications document issued by the Internet Architecture Board.

This setting is not generally used for most mail applications. RFC 822 syntax for parsing addresses effectively enforces a strict definition of a valid email address. If you select this setting, JavaMail will adhere to RFC 822 syntax and reject recipient addresses that do not parse into valid email addresses (as defined by the specification). If you do not select this setting, JavaMail will not adhere to RFC 822 syntax and will accept recipient addresses that do not comply with the specification. By default, this setting is deselected. You can view the RFC 822 specification at the following URL for the World Wide Web Consortium (W3C): http://www.w3.org/Protocols/rfc822/.

Re-Enter Password:

Confirms the password.

Mail From:

Specifies the mail originator.

Mail Store Host:

Specifies the mail account host (or "domain") name.

Mail Store User:

Specifies the user ID of the mail account.

Mail Store Password:

Specifies the password of the mail account.

Re-Enter Password:

Confirms the password.

Mail Store Protocol:

Specifies the protocol to be used when receiving mail.

Mail Debug:

When `true`, JavaMail interaction with mail servers, along with these mail session properties are printed to the `stdout` file.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Example: Configuring mail provider and mail session settings for application clients:

You can configure mail provider and mail session settings. This topic provides the required fields, special cases, and an example.

The purpose of this topic is to help you configure mail provider and mail session settings.

- Required fields:
  - Mail Provider Properties page: name, and at least one protocol provider
  - Mail Session Properties page: name, jndiName, outgoing server and protocol, and/or incoming server and protocol
- Special cases:
  - If you use the ACRCT tool, the password field will be encrypted. You cannot encrypt the password field if you do not use the ACRCT tool.
- Example:

```
<resources.mail:MailProvider xmi:id="builtin_mailprovider" name="Built-in Mail Provider" description="The built-in mail provider">
  <factories xmi:type="resources.mail:MailSession"
             xmi:id="MailSession_1207766754834" name="MailSession"
             jndiName="mail/session" description="Sample mail session" category="Sample"
             mailTransportHost="smtp.coldmail.com" mailTransportUser="transportUser"
             mailTransportPassword="{xor}Lz4sLChvLTs="
             mailFrom="smith@coldmail.com" mailStoreHost="imap.coldmail.com" mailStoreUser="storeUser"
             mailStorePassword="{xor}Lz4sLChvLTs="
             debug="true" strict="true"
             mailTransportProtocol="builtin_smtp" mailStoreProtocol="builtin_imap">
    <propertySet xmi:id="J2EEResourcePropertySet_1207766778585">
      <resourceProperties xmi:id="J2EEResourceProperty_1207766778585" name="key" type="java.lang.String" value="value" required="false"/>
    </propertySet>
  </factories>
  <protocolProviders xmi:id="builtin_smtp" protocol="smtp" classname="com.sun.mail.smtp.SMTPTransport" type="TRANSPORT"/>
  <protocolProviders xmi:id="builtin_pop3" protocol="pop3" classname="com.sun.mail.pop3.POP3Store" type="STORE"/>
  <protocolProviders xmi:id="builtin_imap" protocol="imap" classname="com.sun.mail.imap.IMAPStore" type="STORE"/>
  <protocolProviders xmi:id="builtin_smtps" protocol="smtps" classname="com.sun.mail.smtp.SMTPSSLTransport" type="TRANSPORT"/>
  <protocolProviders xmi:id="builtin_pop3s" protocol="pop3s" classname="com.sun.mail.pop3.POP3SSLStore" type="STORE"/>
  <protocolProviders xmi:id="builtin_imaps" protocol="imaps" classname="com.sun.mail.imap.IMAPSSLStore" type="STORE"/>
</resources.mail:MailProvider>
```

Configuring new mail sessions for application clients:

You can use the Application Client Resource Configuration Tool (ACRCT) to configure new mail sessions for your application client.

Before you begin

During this task, you configure new mail sessions for your application client. The mail sessions are associated with the pre-configured default mail provider supplied by the product.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT) and open the EAR file. The EAR file contents are displayed in a tree view.
2. Select the JAR file in which you want to configure the new JavaMail session.
3. Expand the JAR file to view its contents.
4. Click **Mail Providers > Mail Provider > Mail Sessions**. Complete one of the following actions:
   - Right click the Mail Sessions folder and select **New Factory**.
   - Click **Edit > New** on the menu bar.
5. Configure the Mail Session properties in the displayed fields.
6. Click **OK**.
7. Click **File > Save** on the menu bar to save your changes.

Configuring new URL providers for application clients:

You can create URL providers and URLs for your client application using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

During this task, you create URL providers and URLs for your client application. In a separate administrative task, you must install the Java code for the required URL provider on the client machine on which the client application resides.

About this task

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT).
2. Open the EAR file for which you want to configure the new URL provider. The EAR file contents display in a tree view.
3. Select the JAR file in which you want to configure the new URL provider from the tree.
4. Expand the JAR file to view the contents.
5. Click the folder called URL Providers. Complete one of the following actions:
   - Right click the folder and select **New**.
   - Click **Edit > New** on the menu bar.
6. Configure the URL provider properties in the resulting property dialog.
7. Click **OK**.
8. Click **File > Save** on the menu bar to save your changes.

Example

- Configuring URL and URL provider settings for application clients

This code example illustrates how to configure URL and URL provider settings for application clients:

```
<resources.url:URLProvider xmi:id="URLProvider_1" name="urlProvider:name"
description="urlProvider:description"
streamHandlerClassName="urlProvider:streamHandlerClass"
protocol="urlProvider:protocol">
<classpath>urlProvider:classpath</classpath>
<factories xmi:type="resources.url:URL" xmi:id="URL_1" name="urlFactory:name"
jndiName="urlFactory:jndiName" description="urlFactory:description"
spec="urlFactory:url">
<propertySet xmi:id="J2EEResourcePropertySet_18">
<resourceProperties xmi:id="J2EEResourceProperty_20" name="urlFactory:customName"
value="urlFactory:customValue"/>
</propertySet>
```

```
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_19">
<resourceProperties xmi:id="J2EEResourceProperty_21" name="urlProvider:customName"
value="urlProvider:customValue"/>
</propertySet>
</resources.url:URLProvider>
```

- Required fields:
  - URL Properties page: name, jndiName, url
  - URL Provider Properties page: name

URLs for application clients:

A *Uniform Resource Locator* (URL) is an identifier that points to an electronically accessible resource, such as a directory file on a machine in a network, or a document stored in a database.

URLs appear in the format *scheme*:*scheme_information*.

You can represent a *scheme* as `http`, `ftp`, `file`, or another term that identifies the type of resource and the mechanism by which you can access the resource.

In a web browser location or address box, a URL for a file available using HyperText Transfer Protocol (HTTP) starts with `http:`. An example is `http://www.ibm.com`. Files available using File Transfer Protocol (FTP) start with `ftp:`. Files available locally start with `file:`.

The *scheme_information* commonly identifies the Internet machine making a resource available, the path to that resource, and the resource name. The scheme_information for HTTP, FTP and File generally starts with two slashes (//), then provides the Internet address separated from the resource path name with one slash (/). For example,

`http://www.ibm.com/software/webservers/appserv/library.html`.

For HTTP and FTP, the path name ends in a slash when the URL points to a directory. In such cases, the server generally returns the default index for the directory.

URL providers for the Application Client Resource Configuration Tool:

A URL provider implements the function for a particular URL protocol, such as HyperText Transfer Protocol (HTTP). This provider, comprised of a pair of classes, extends the java.net.URLStreamHandler and java.net.URLConnection classes.

Configuring URL providers and sessions using the Application Client Resource Configuration Tool:

You can edit the configurations of URL providers and URLs to be used by your application clients using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

Use the Application Client Resource Configuration Tool (ACRCT) to edit the configurations of URL providers and URLs to be used by your application clients.

About this task

Procedure

1. Start the ACRCT.
2. Open an EAR file.

3. Locate the URL objects in the tree that displays. For example, if your file contains URL providers and URLs, expand **Resources > *application* > .jar > URL Providers > *url_provider_instance***
   where *url_provider_instance* is a particular URL provider.
4. If you expand the tree further, you will also see the URLs folders containing the URL instances for each URL provider instance.

URL settings for application clients:

Use this page to implement the function for a particular URL protocol, such as Hyper Text Transfer Protocol (HTTP).

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **URL Providers** > *URL provider instance*. Right-click **URLs** and click **New**. The following fields appear on the **General** tab.

This provider, comprised of classes, extends the `java.net.URLStreamHandler` and `java.net.URLConnection` classes.

Name:

The administrative name for the URL.

Description:

This is an optional description of the URL for your administrative records.

JNDI Name:

The application client run time uses this field to retrieve configuration information.

URL:

A Uniform Resource Locator (URL) name that points to an Internet or intranet resource. For example: `http://www.ibm.com`.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

URL provider settings for application clients:

Use this page create new URL providers.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file. Right click **URL Providers**, and click **New**. The following fields appear on the **General** tab.

A URL provider implements the function for a particular URL protocol, such as Hyper Text Transfer Protocol (HTTP). This provider, comprised of classes, extends the `java.net.URLStreamHandler` and `java.net.URLConnection` classes.

Name:

Administrative name for the URL.

Description:

Optional description of the URL, for your administrative records.

Class Path:

A list of paths or JAR file names which together form the location for the resource provider classes.

Protocol:

Protocol supported by this stream handler. For example, `nntp`, `smtp`, `ftp`, and so on.

To use the default protocol, leave this field blank.

Stream handler class:

Fully qualified name of a User-defined Java class that extends the `java.net.URLStreamHandler` for a particular URL protocol, such as FTP.

To use the default stream handler, leave this field blank.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Example: Configuring URL and URL provider settings for application clients:

You can configure URL and URL provider settings. This topic provides the required fields and an example.

The purpose of this article is to help you to configure URL and URL provider settings.

- Required fields:
  - URL Properties page: name, jndiName, url
  - URL Provider Properties page: name
- Example:

```
<resources.url:URLProvider xmi:id="URLProvider_1" name="urlProvider:name"
description="urlProvider:description"
streamHandlerClassName="urlProvider:streamHandlerClass"
protocol="urlProvider:protocol">
<classpath>urlProvider:classpath</classpath>
<factories xmi:type="resources.url:URL" xmi:id="URL_1" name="urlFactory:name"
jndiName="urlFactory:jndiName" description="urlFactory:description"
spec="urlFactory:url">
<propertySet xmi:id="J2EEResourcePropertySet_18">
<resourceProperties xmi:id="J2EEResourceProperty_20" name="urlFactory:customName"
value="urlFactory:customValue"/>
</propertySet>
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_19">
```

```
<resourceProperties xmi:id="J2EEResourceProperty_21" name="urlProvider:customName"
value="urlProvider:customValue"/>
</propertySet>
</resources.url:URLProvider>
```

Configuring new URLs with the Application Client Resource Configuration Tool:

You can use URLs for your client application using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

During this task, you create URLs for your client application.

About this task

Procedure

1. Click the URL provider for which you want to create a URL in the tree. Complete one of the following:
   - Configure a new URL provider.
   - Click an existing URL provider.
2. Expand the URL provider to view the URLs folder.
3. Click the URL folder. Complete one of the following actions:
   - Right click the folder and click **New**.
   - Click **Edit -> New** on the menu bar.
4. Configure the URL properties in the displayed fields.
5. Click **OK** when you finish.
6. Click **File > Save** in the menu bar to save your changes.

Configuring Java messaging client resources:

To configure Java messaging client resources, you create new JMS provider configurations for your application client. The application client can use a messaging service through the Java Message Service APIs. A JMS provider provides two kinds of J2EE factories. One is a *JMS connection factory*, and the other is a *JMS destination factory*.

Before you begin

In a separate administrative task, install the Java Message Service (JMS) client on the client machine where the application client resides. The messaging product vendor must provide an implementation of the JMS client. For more information, see your messaging product documentation.

Attention: When completing this task, you can either create a new messaging provider, or you can use an existing one.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT).
2. Open the EAR file for which you want to configure the new JMS provider. The EAR file contents are in the displayed tree view.
3. Select the JAR file in which you want to configure the new JMS provider from the tree.
4. Expand the JAR file to view its contents.
5. Optionally right-click **Messaging Providers** and select **New**, if you want to create and use a new messaging provider.
6. Configure the JMS provider properties in the resulting property dialog.

7. Click **OK**.
8. Click **File > Save**.

Asynchronous messaging in WebSphere Application Server using JMS:

WebSphere Application Server supports asynchronous messaging as a method of communication based on the Java Message Service (JMS) programming interface. The JMS interface provides a common way for Java programs (clients and Java Platform, Enterprise Edition (Java EE) applications) to create, send, receive, and read asynchronous requests as JMS messages.

This topic provides a generic overview of asynchronous messaging using the JMS support provided by WebSphere Application Server.

The base support for asynchronous messaging using the JMS API provides the common set of JMS interfaces and associated semantics that define how a JMS client can access the facilities of a JMS provider. This support enables WebSphere product Java EE applications, as JMS clients, to exchange messages asynchronously with other JMS clients, by using JMS destinations (queues or topics). A Java EE application can use JMS queue destinations for point-to-point messaging and JMS topic destinations for publish and subscribe messaging. A Java EE application can explicitly poll for messages on a destination, and then retrieve messages for processing by business logic beans (enterprise beans).

With the base JMS and XA support, the Java EE application uses standard JMS calls to process messages, including any responses or outbound messaging. An enterprise bean can handle responses acting as a sender bean, or within the enterprise bean that receives the incoming messages. Optionally, this process can use two-phase commit within the scope of a transaction. This level of function for asynchronous messaging is called *bean-managed messaging*, and gives an enterprise bean complete control over the messaging infrastructure, for example, connection and session pool management. The common container has no role in bean-managed messaging.

WebSphere Application Server also supports automatic asynchronous messaging using message-driven beans (a type of enterprise bean defined in the Enterprise JavaBeans (EJB) 2.0 specification) and JMS listeners (part of the JMS application server facilities). Messages are automatically retrieved from JMS destinations, optionally within a transaction, then sent to the message-driven bean in a Java EE application, without the application having to explicitly poll JMS destinations.

Java Message Service providers for clients:

Client applications can use messaging resources from three main types of Java Message Service (JMS) providers in WebSphere Application Server: The WebSphere Application Server default messaging provider (which uses service integration as the provider), the WebSphere MQ messaging provider (which uses your WebSphere MQ system as the provider) and third-party messaging providers (which use another company's product as the provider).

IBM WebSphere Application Server supports asynchronous messaging through the use of a JMS provider and its related messaging system. JMS providers must conform to the JMS specification version 1.1. To use message-driven beans the JMS provider must support the optional Application Server Facility (ASF) function defined within that specification, or support an inbound resource adapter as defined in the JCA specification version 1.5.

Default messaging provider

If you mainly want to use messaging between applications in WebSphere Application Server, perhaps with some interaction with a WebSphere MQ system, the default messaging provider is the natural choice. This provider is based on service integration technologies and is fully integrated with the WebSphere Application Server runtime environment.

WebSphere MQ messaging provider

If your business also uses WebSphere MQ, and you want to integrate WebSphere Application

Server messaging applications into a predominately WebSphere MQ network, choose the WebSphere MQ messaging provider, which allows you to define resources for connecting to any queue manager on the WebSphere MQ network.

Third-party messaging provider
You can configure any third-party messaging provider that supports the JMS Version 1.1 unified connection factory. You might want to do this, for example, because of existing investments.

Note: In WebSphere Application Server 7.0, the Version 5 default messaging provider is deprecated.

Note: For backwards compatibility with earlier releases, WebSphere Application Server Version 7 supports the (deprecated) Version 5 default messaging provider and the Version 6 WebSphere MQ messaging provider. This support enables your applications that still use these resources to communicate with Version 5 and Version 6 nodes in Version 7 mixed cells.

WebSphere applications can use messaging resources provided by any of these JMS providers. However the choice of provider is most often dictated by requirements to use or integrate with an existing messaging system. For example, you may already have a messaging infrastructure based on WebSphere MQ. In this case you may either connect directly using the included support for WebSphere MQ as a JMS provider, or configure a service integration bus with links to a WebSphere MQ network and then access the bus through the default messaging provider.

Configuring new JMS providers with the Application Client Resource Configuration Tool:

You can create new Java Message Service (JMS) provider configurations for the Application Client. The Application Client makes use of a messaging service through the JMS interfaces.

About this task

During this task, you create new Java Message Service (JMS) provider configurations for the Application Client. The Application Client makes use of a messaging service through the JMS interfaces. A JMS provider provides two kinds of Java Platform, Enterprise Edition (Java EE) resources. One is a JMS connection factory, and the other is a JMS destination.

In a separate administrative task, you must install the JMS client on the client machine where your particular application client resides. The messaging product vendor must provide an implementation of the JMS client. For more information, see your messaging product documentation.

Procedure

1. Start the Application Client Resource Configuration Tool and open the EAR file for which you want to configure the new JMS provider. The EAR file contents are displayed in a tree view.
2. From the tree, select the JAR file in which you want to configure the new JMS provider.
3. Expand the JAR file to view its contents.
4. Right-click **Messaging Providers**. Complete one of the following actions:
   - Right click the folder and select **New**.
   - On the menu bar, click **Edit > New**.
5. In the resulting property dialog, configure the JMS provider properties.
6. Click **OK** when finished.
7. Click **File > Save** on the menu bar to save your changes.

Example

The following code example illustrates how to configure JMS Provider, JMS Connection Factory and JMS Destination settings for application clients.

```
<resources.jms:JMSProvider xmi:id="JMSProvider_3" name="genericJMSProvider:name"
description="genericJMSProvider:description"
externalInitialContextFactory="genericJMSProvider:contextFactoryClass"
externalProviderURL="genericJMSProvider:providerUrl">
<classpath>genericJMSProvider:classpath</classpath>
<factories xmi:type="resources.jms:GenericJMSDestination"
xmi:id="GenericJMSDestination_1" name="jmsDestination:name"
jndiName="jmsDestination:jndiName" description="jmsDestination:description"
externalJNDIName="jmsDestination:externalJndiName" type="QUEUE">
<propertySet xmi:id="J2EEResourcePropertySet_15">
<resourceProperties xmi:id="J2EEResourceProperty_17" name="jmsDestination:customName"
value="jmsDestination:customValue"/>
</propertySet>
</factories>
<factories xmi:type="resources.jms:GenericJMSConnectionFactory"
xmi:id="GenericJMSConnectionFactory_1" name="jmsCF:name" jndiName="jmsCF:jndiName"
description="jmsCF:description" userID="jmsCF:user" password="{xor}NTIsHBllMT4yOg=="
externalJNDIName="jmsCF:externalJndiName" type="QUEUE">
<propertySet xmi:id="J2EEResourcePropertySet_16">
<resourceProperties xmi:id="J2EEResourceProperty_18" name="jmsCF:customName"
value="jmsCF:customValue"/>
</propertySet>
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_17">
<resourceProperties xmi:id="J2EEResourceProperty_19"
name="genericJMSProvider:customName" value="genericJMSProvider:customValue"/>
</propertySet>
</resources.jms:JMSProvider>
```

Required fields include:
- JMS Provider Properties page: name, and at least one protocol provider
- JMS Connection Factory Properties page: name, jndiName, destination type
- JMS Destination Properties page: name, jndiName, destination type

Special cases:
- The destination type must be QUEUE, or TOPIC.

JMS provider settings for application clients:

Use this page to configure properties of the Java Message Service (JMS) provider, if you want to use a JMS provider other than the default messaging provider or the WebSphere MQ as a JMS provider.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file. Right click **Messaging Providers**, and click **New**. The following fields appear on the **General** tab.

Name:

The name by which the JMS provider is known for administrative purposes.

Data type String

Description:

A description of the JMS provider, for administrative purposes.

Data type String

Class Path:

A list of paths or `.jar` file names which together form the location for the resource provider classes.

Context factory class:

The Java class name of the initial context factory for the JMS provider.

For example, for an LDAP service provider the value has the form: `com.sun.jndi.ldap.LdapCtxFactory`.

Data type String

Provider URL:

The JMS provider URL for external JNDI lookups.

For example, an LDAP URL for a JMS provider has the form: `ldap://hostname.company.com/contextName`.

Data type String

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Default Provider connection factory settings:

Use this panel to view or change the configuration properties of the selected JMS connection factory for use with the internal product Java Message Service (JMS) provider that is installed with WebSphere Application Server. These configuration properties control how connections are created between the JMS provider and the service integration bus that it uses

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Default Provider**. Right-click **Connection Factories** and click **New**. The following fields appear on the **General** tab.

Settings that have a default value display the appropriate value. Any settings that have fixed values have a drop down menu.

Name:

The name of the connection factory.

Data type String

Description:

A description of this connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String

JNDI Name:

The JNDI name that is used to match this Resource Adapter connection factory definition to the deployment descriptor. This entry is a `resource-ref` name.

Data type String

User Name:

The **User Name** used with the **Password** property for connecting to an application.

If you specify a value for the **User Name** property, you must also specify a value for the **Password** property.

The connection factory **User ID** and **Password** properties are used if the calling application does not provide a `userid` and `password` explicitly. If a user name and password are specified, then an authentication alias is created for the factory where the password is encrypted.

Data type String

Password:

The password used to authenticate connection to an application.

If you specify a value for the **User Name** property, you must also specify a value for the **Password** property.

Data type String

Re-Enter Password:

Confirms the password.

Bus Name:

The name of the bus to which the connection factory connects.

Data type String

Client Identifier:

The name of the client. Required for durable topic subscriptions.

Data type String

Nonpersistent Messaging Reliability:

The reliability applied to nonpersistent JMS messages sent using this connection factory.

If you want different reliability delivery options for individual JMS destinations, you can set this property to **As bus** destination. The reliability is then defined by the Reliability property of the bus destination to which the JMS destination is assigned.

Default
ReliablePersistent

Range

None There is no message reliability for nonpersistent messages. If a nonpersistent message cannot be delivered, it is discarded.

Best effort nonpersistent
Messages are never written to disk, and are thrown away if memory cache overruns.

Express nonpersistent
Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts.

Reliable nonpersistent
Messages can be lost if a messaging engine fails, and can be lost under normal operating conditions.

Reliable persistent
Messages can be lost if a messaging engine fails, but are not lost under normal operating conditions.

Assured persistent
Highest degree of reliability where assured message delivery is supported.

As Bus destination
Use the delivery option configured for the bus destination.

Persistent Message Reliability:

The reliability applied to persistent JMS messages sent using this connection factory.

If you want different reliability delivery options for individual JMS destinations, you can set this property to **As bus destination**. The reliability is then defined by the Reliability property of the bus destination to which the JMS destination is assigned.

Default
ReliablePersistent

Range

None There is no message reliability for nonpersistent messages. If a nonpersistent message cannot be delivered, it is discarded.

Best effort nonpersistent
Messages are never written to disk, and are thrown away if memory cache overruns.

Express nonpersistent
Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts.

Reliable nonpersistent
Messages can be lost if a messaging engine fails, and can be lost under normal operating conditions.

Reliable persistent
Messages can be lost if a messaging engine fails, but are not lost under normal operating conditions.

Assured persistent
Highest degree of reliability where assured message delivery is supported.

As Bus destination
Use the delivery option configured for the bus destination.

Durable Subscription Home:

The name of the durable subscription home.

Data type String

Share durable subscriptions:

Controls whether or not durable subscriptions are shared across connections with members of a server cluster.

Normally, only one session at a time can have a TopicSubscriber for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers.

Data type Selection list
Default In cluster
Range

In cluster
Allows sharing of durable subscriptions when connections are made from within a server cluster.

Always shared
Durable subscriptions can be shared across connections.

Never shared
Durable subscriptions are never shared across connections.

Read Ahead:

Controls the read-ahead optimization during message delivery.

Default	Default
Range	Default, AlwaysOn and AlwaysOff

Target:

The name of the Workload Manager target group containing the messaging engine.

Data type	String

Target Type:

The type of Workload Manager target group that contains the messaging engine.

Default	BusMember
Range	BusMember, Custom, ME

Target Significance:

The priority of significance for the target specified.

Default	Preferred
Range	Preferred, Required

Target Inbound Transport Chain:

The name of the protocol that resolves to a group of messaging engines.

Data type	String

Provider Endpoints:

The list of comma separated endpoints used to connect to a bootstrap server.

Type a comma-separated list of endpoint triplets with the syntax: host:port:protocol.

Example	`merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging` where `BootstrapBasicMessaging` corresponds to the remote protocol InboundBasicMessaging (JFAP-TCP/IP).
Default	• If the host name is not specified, then the default `localhost` is used as a default value. • If the port number is not specified, then `7276` is used as a default value. • If the chain name is not specified, a predefined chain, such as `BootstrapBasicMessaging`, is used as a default value.

Connection Proximity:

The proximity that the messaging engine should have to the requester.

Default Bus
Range Bus, Host, Cluster, Server

Temporary Queue Name Prefix:

The prefix to apply to the names of temporary queues. This name is a maximum of 12 characters.

Data type String

Temporary Topic Name Prefix:

The prefix to apply to the names of temporary topics. This name is a maximum of 12 characters.

Data type String

Default Provider queue connection factory settings:

Use this panel to view or change the configuration properties of the selected JMS queue connection factory for use with the internal product Java Message Service (JMS) provider that is installed with WebSphere Application Server. These configuration properties control how connections are created between the JMS provider and the service integration bus that it uses

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Default Provider**. Right-click **Queue Connection Factories** and click **New**. The following fields appear on the **General** tab.

Settings that have a default value display the appropriate value. Any settings that have fixed values have a drop down menu.

Name:

The name of the queue connection factory.

Data type String

Description:

A description of this queue connection factory for administrative purposes within WebSphere Application Server.

Data type String

JNDI Name:

The JNDI name that is used to match this queue connection factory definition to the deployment descriptor. This entry is a `resource-ref` name.

Data type String

User Name:

The **User Name** used, with the **Password** property, for authentication if the calling application does not provide a `userid` and `password` explicitly. If this field is used, then the Properties field `UserName` is ignored.

If you specify a value for the **User Name** property, you must also specify a value for the **Password** property.

The connection factory **User Name** and **Password** properties are used if the calling application does not provide a `userid` and `password` explicitly. If a user name and password are specified, then an authentication alias is created for the factory where the password is encrypted.

Data type String

Password:

The password used to create an encrypted. If you complete this field, then the Password field in the Properties box is ignored.

If you specify a value for the **User Name** property, you must also specify a value for the **Password** property.

Data type String

Re-Enter Password:

Confirms the password.

Bus Name:

The name of the bus to which the queue connection factory connects.

Data type String

Client Identifier:

The client identifier. Required for durable topic subscriptions.

Data type String

Nonpersistent Messaging Reliability:

The reliability applied to nonpersistent JMS messages sent using this connection factory.

If you want different reliability delivery options for individual JMS destinations, you can set this property to **As bus** destination. The reliability is then defined by the Reliability property of the bus destination to which the JMS destination is assigned.

Default ReliablePersistent

Range

None There is no message reliability for nonpersistent messages. If a nonpersistent message cannot be delivered, it is discarded.

Best effort nonpersistent
Messages are never written to disk, and are thrown away if memory cache overruns.

Express nonpersistent
Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts.

Reliable nonpersistent
Messages can be lost if a messaging engine fails, and can be lost under normal operating conditions.

Reliable persistent
Messages can be lost if a messaging engine fails, but are not lost under normal operating conditions.

Assured persistent
Highest degree of reliability where assured message delivery is supported.

As Bus destination
Use the delivery option configured for the bus destination.

Persistent Message Reliability:

The reliability applied to persistent JMS messages sent using this connection factory.

If you want different reliability delivery options for individual JMS destinations, you can set this property to **As bus destination**. The reliability is then defined by the Reliability property of the bus destination to which the JMS destination is assigned.

Default ReliablePersistent

Range

None There is no message reliability for nonpersistent messages. If a nonpersistent message cannot be delivered, it is discarded.

Best effort nonpersistent
Messages are never written to disk, and are thrown away if memory cache overruns.

Express nonpersistent
Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts.

Reliable nonpersistent
Messages can be lost if a messaging engine fails, and can be lost under normal operating conditions.

Reliable persistent
Messages can be lost if a messaging engine fails, but are not lost under normal operating conditions.

Assured persistent
Highest degree of reliability where assured message delivery is supported.

As Bus destination
Use the delivery option configured for the bus destination.

Read Ahead:

Controls the read-ahead optimization during message delivery.

Default Default
Range Default, AlwaysOn and AlwaysOff

Target:

The name of the Workload Manager target group containing the messaging engine.

Data type String

Target Type:

The type of Workload Manager target group that contains the messaging engine.

Default BusMember
Range BusMember, Custom, Destination, ME

Target Significance:

The priority of significance for the target specified.

Default Preferred
Range Preferred, Required

Target Inbound Transport Chain:

The name of the protocol that resolves to a group of messaging engines.

Data type String

Provider Endpoints:

The list of comma separated endpoints used to connect to a bootstrap server.

Type a comma-separated list of endpoint triplets with the syntax: host:port:protocol.

Example `localhost:7777:BootstrapBasicMessaging`

where

`BootstrapBasicMessaging` corresponds to the remote protocol InboundBasicMessaging (JFAP-TCP/IP).

Default

- If the host name is not specified, then the default `localhost` is used as a default value.
- If the port number is not specified, then `7276` is used as a default value.
- If the chain name is not specified, a predefined chain, such as `BootstrapBasicMessaging`, is used as a default value.

Connection Proximity:

The proximity that the messaging engine should have to the requester.

Default Bus, Cluster, Server
Range Bus, Host

Temporary Queue Name Prefix:

The prefix to apply to the names of temporary queues. This name is a maximum of 12 characters.

Data type String

Default Provider topic connection factory settings:

Use this panel to view or change the configuration properties of the selected JMS topic connection factory for use with the internal product Java Message Service (JMS) provider that is installed with WebSphere Application Server. These configuration properties control how connections are created between the JMS provider and the service integration bus that it uses.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Default Provider**. Right-click **Topic Connection Factories** and click **New**. The following fields appear on the **General** tab.

Settings that have a default value display that appropriate value. Any settings that have fixed values have a drop down menu.

Name:

The name of the topic connection factory.

Data type String

Description:

A description of this topic connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String

JNDI Name:

The JNDI name that is used to match this topic connection factory definition to the deployment descriptor. This entry is a `resource-ref` name.

Data type String

User Name:

The **User Name** used, with the **Password** property, for authentication if the calling application does not provide a `userid` and `password` explicitly. If this field is used, then the Properties field `UserName` is ignored.

If you specify a value for the **User Name** property, you must also specify a value for the **Password** property.

The connection factory **User Name** and **Password** properties are used if the calling application does not provide a `userid` and `password` explicitly. If a user name and password are specified, then an authentication alias is created for the factory where the password is encrypted.

Data type String

Password:

The password used to create an encrypted. If you complete this field, then the Password field in the Properties box is ignored.

If you specify a value for the **User Name** property, you must also specify a value for the **Password** property.

Data type String

Re-Enter Password:

Confirms the password.

Bus Name:

The name of the bus to which the topic connection factory connects.

Data type String

Client Identifier:

The name of the client. This field is required for durable topic subscriptions.

Data type String

Nonpersistent Messaging Reliability:

The reliability applied to nonpersistent JMS messages sent using this connection factory.

If you want different reliability delivery options for individual JMS destinations, you can set this property to **As bus** destination. The reliability is then defined by the Reliability property of the bus destination to which the JMS destination is assigned.

Default ReliablePersistent

Range

None There is no message reliability for nonpersistent messages. If a nonpersistent message cannot be delivered, it is discarded.

Best effort nonpersistent
Messages are never written to disk, and are thrown away if memory cache overruns.

Express nonpersistent
Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts.

Reliable nonpersistent
Messages can be lost if a messaging engine fails, and can be lost under normal operating conditions.

Reliable persistent
Messages can be lost if a messaging engine fails, but are not lost under normal operating conditions.

Assured persistent
Highest degree of reliability where assured message delivery is supported.

As Bus destination
Use the delivery option configured for the bus destination.

Persistent Message Reliability:

The reliability applied to persistent JMS messages sent using this connection factory.

If you want different reliability delivery options for individual JMS destinations, you can set this property to **As bus destination**. The reliability is then defined by the Reliability property of the bus destination to which the JMS destination is assigned.

Default ReliablePersistent

Range

None There is no message reliability for nonpersistent messages. If a nonpersistent message cannot be delivered, it is discarded.

Best effort nonpersistent
Messages are never written to disk, and are thrown away if memory cache overruns.

Express nonpersistent
Messages are written asynchronously to persistent storage if memory cache overruns, but are not kept over server restarts.

Reliable nonpersistent
Messages can be lost if a messaging engine fails, and can be lost under normal operating conditions.

Reliable persistent
Messages can be lost if a messaging engine fails, but are not lost under normal operating conditions.

Assured persistent
Highest degree of reliability where assured message delivery is supported.

As Bus destination
Use the delivery option configured for the bus destination.

Durable Subscription Home:

The name of the durable subscription home.

Data type String

Share durable subscriptions:

Controls whether or not durable subscriptions are shared across connections with members of a server cluster.

Normally, only one session at a time can have a TopicSubscriber for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers.

Data type Selection list
Default In cluster
Range

In cluster
Allows sharing of durable subscriptions when connections are made from within a server cluster.

Always shared
Durable subscriptions can be shared across connections.

Never shared
Durable subscriptions are never shared across connections.

Read Ahead:

Controls the read-ahead optimization during message delivery.

Default	Default
Range	Default, AlwaysOn and AlwaysOff

Target:

The name of the Workload Manager target group containing the messaging engine.

Data type	String

Target Type:

The type of Workload Manager target group that contains the messaging engine.

Default	BusMember
Range	BusMember, Custom, ME

Target Significance:

The priority of significance for the target specified.

Default	Preferred
Range	Preferred, Required

Target Inbound Transport Chain:

The name of the protocol that resolves to a group of messaging engines.

Data type	String

Provider Endpoints:

The list of comma separated endpoints used to connect to a bootstrap server.

Type a comma-separated list of endpoint triplets with the syntax: host:port:protocol.

Example	`localhost:7777:BootstrapBasicMessaging` where `BootstrapBasicMessaging` corresponds to the remote protocol InboundBasicMessaging (JFAP-TCP/IP).
Default	• If the host name is not specified, then the default `localhost` is used as a default value. • If the port number is not specified, then `7276` is used as a default value. • If the chain name is not specified, a predefined chain, such as `BootstrapBasicMessaging`, is used as a default value.

Connection Proximity:

The proximity that the messaging engine should have to the requester.

Default Bus
Range Bus, Host, Cluster, Server

Temporary Topic Name Prefix:

The prefix to apply to the names of temporary topics. This name is a maximum of 12 characters.

Data type String

Default Provider queue destination settings:

Use this panel to view or change the configuration properties of the selected JMS queue destination for use with the internal product Java Message Service (JMS) provider that is installed with WebSphere Application Server.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Default Provider**. Right-click **Queue Destinations**. Click **New**. The following fields appear on the **General** tab.

Name:

The name of the queue destination factory. You must complete this field.

Data type String

Description:

A description of this queue destination for administrative purposes within WebSphere Application Server.

Data type String

JNDI Name:

The JNDI name used to match this definition to a deployment descriptor `resource-env-ref` name.

Data type String

Queue Name:

The name of the queue.

Data type String

Delivery Mode:

The delivery mode for messages sent to this destination.

Data type String
Range Application, Persistent or NonPersistent

Default Application

Time to Live:

The default length of time from its dispatch time that a message sent to this destination should be retained by the system, where **0** indicates that time to live value does not expire. Value from the producer is used if the Time to Live field is not completed.

Data type Integer
Units Milliseconds

Priority:

The priority for messages sent to this destination. The value from the producer is used if not completed.

Data type Integer
Range 0 to 9 with **0** as the lowest priority and **9** as the highest priority

Read Ahead:

Used to control read-ahead optimization during message delivery.

Data type String
Range AsConnection, AlwaysOn and AlwaysOff
Default AsConnection

Default Provider topic destination settings:

Use this panel to view or change the configuration properties of the selected JMS topic destination for use with the internal product Java Message Service (JMS) provider that is installed with WebSphere Application Server.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Default Provider**. Right-click **Topic Destinations**, and click **New**. The following fields appear on the **General** tab.

Name:

The name of the topic destination entry.

Data type String

Description:

A description of the entry.

Data type String

JNDI Name:

The JNDI name used to match this definition to a deployment descriptor `resource-env-ref` name.

Data type String

Topic Space:

The name of the topic space. This field is required.

Data type String
Default DEFAULT_TOPIC_SPACE

Topic Name:

The name of the topic. This field is required.

Data type String

Delivery Mode:

The default mode for messages sent to this destination.

Data type String
Range Application, Persistent or NonPersistent
Default Application

Time to Live:

The default length of time from its dispatch time that a message sent to this destination should be retained by the system, where **0** indicates that time to live value does not expire. Value from the producer is used if not completed.

Data type Long
Units Milliseconds

Priority:

The priority for messages sent to this destination. Value from producer is used if not completed.

Data type Integer
Range 0 to 9 with **0** as the lowest priority and **9** as the highest priority

Read Ahead:

Used to control read-ahead optimization during message delivery.

Data type String
Range AsConnection, AlwaysOn and AlwaysOff
Default AsConnection

V5 default messaging provider queue connection factory settings for application clients:

Use this panel to browse or change the configuration properties of the selected JMS queue connection factory for point-to-point messaging for use by WebSphere Application Server version 5 applications. These configuration properties control how connections are created between the JMS provider and the default messaging system that it uses.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Provider** > **Version 5 Default Provider**. Right-click **Queue Connection Factories** and click **New**. The following fields appear on the **General** tab.

A queue connection factory is used to create JMS connections to queue destinations. The queue connection factory is created by the internal WebSphere Application Server product JMS provider. A V5 default messaging provider queue connection factory has the following properties:

Name:

The name by which this queue connection factory is known for administrative purposes within IBM WebSphere Application Server. The name must be unique within the JMS connection factories across the WebSphere administrative domain.

Data type String

Description:

A description of this connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String
Default Null

JNDI Name:

The application client run time uses this field to retrieve configuration information.

User ID:

The User ID used, with the **Password** property, for authentication if the calling application does not provide a `userid` and `password` explicitly.

If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.

The connection factory **User ID** and **Password** properties are used if the calling application does not provide a User ID and password explicitly, for example, if the calling application uses the method createQueueConnection(). The JMS client flows the `userid` and `password` to the JMS server.

Data type String

Password:

The password used, with the **User ID** property, for authentication if the calling application does not provide a `userid` and `password` explicitly.

If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.

Re-Enter Password:

Confirms the password.

Node:

The WebSphere node name of the administrative node where the JMS server runs for this connection factory. Connections created by this factory connect to that JMS server.

Data type String

Application Server:

Enter the name of the application server. This name is not the host name of the machine, but the name of the configured application server.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

V5 default messaging provider topic connection factory settings for application clients:

Use this panel to view or change the configuration properties of the selected topic connection factory for use with the internal product Java Message Service (JMS) provider. These configuration properties control how connections are created between the JMS provider and the messaging system that it uses.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Version 5 Default Provider**. Right click **Topic Connection Factories** and click **New**. The following fields appear on the **General** tab.

A V5 default messaging provider topic connection factory has the following properties.

Name:

The name by which this queue connection factory is known for administrative purposes within WebSphere Application Server. The name must be unique within the JMS connection factories across the WebSphere Application Server administrative domain.

Data type String

Description:

A description of this topic connection factory for administrative purposes within WebSphere Application Server.

Data type String

JNDI Name:

The application client run time uses this field to retrieve configuration information.

User ID:

The user ID used, with the **Password** property, for authentication if the calling application does not provide a `userid` and `password` explicitly.

If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.

The connection factory **User ID** and **Password** properties are used if the calling application does not provide a `userid` and `password` explicitly, for example, if the calling application uses the method createTopicConnection(). The JMS client flows the `userid` and `password` to the JMS server.

Data type	String

Password:

The password used, with the **User ID** property, for authentication if the calling application does not provide a `userid` and `password` explicitly.

If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.

Data type	String

Re-Enter Password:

Confirms the password.

Node:

The WebSphere Application Server node name of the administrative node where the JMS server runs for this connection factory. Connections created by this factory connect to that JMS server.

Data type	Enum
Range	Pull-down list of nodes in the WebSphere Application Server administrative domain.

Application Server:

Enter the name of the application server. This name is not the host name of the machine, but the name of the configured application server.

Port:

Which of the two ports that connections use to connect to the JMS Server. The QUEUED port is for full-function JMS publish/subscribe support, the DIRECT port is for nonpersistent, nontransactional, nondurable subscriptions only.

Note: Message-driven beans cannot use the direct listener port for publish or subscribe support. Therefore, any topic connection factory configured with the Port set to `Direct` cannot be used with message-driven beans.

Data type	Enum
Default	QUEUED

Range

QUEUED
The listener port used for full-function JMS compliant, publish or subscribe support.

DIRECT
The listener port used for direct TCP/IP connection (nontransactional, nonpersistent, and nondurable subscriptions only) for publish or subscribe support.

The TCP/IP port numbers for these ports are defined on the product internal JMS server.

Client ID:

The JMS client identifier used for connections to the MQSeries® queue manager.

Data type String

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

V5 default messaging provider queue destination settings for application clients:

Use this panel to view or change the configuration properties of the selected queue destination for use with product Java Message Service (JMS) provider.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Version 5 Default Provider**. Right click **Queue Destinations** and click **New**. The following fields are displayed on the **General** tab.

A queue destination is used to configure the properties of a JMS queue. A V5 default messaging provider queue destination has the following properties.

Name:

The name by which the queue is known for administrative purposes within WebSphere Application Server.

Data type String

Description:

A description of the queue, for administrative purposes.

Data type String

JNDI Name:

The application client run time uses this field to retrieve configuration information.

Persistence:

Whether all messages sent to the destination are persistent, nonpersistent, or have their persistence defined by the application.

Data type Enum
Default APPLICATION_DEFINED
Range
Application defined
Messages on the destination have their persistence defined by the application that put them onto the queue.
Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.
Persistent
Messages on the destination are persistent.
Nonpersistent
Messages on the destination are not persistent.

Priority:

Whether the message priority for this destination is defined by the application or the **Specified priority** property.

Data type Enum
Default APPLICATION_DEFINED
Range
Application defined
The priority of messages on this destination is defined by the application that put them onto the destination.
Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.
Specified
The priority of messages on this destination is defined by the **Specified priority** property. *If you select this option, you must define a priority on the **Specified priority** property.*

Specified Priority:

If the **Priority** property is set to `Specified`, type here the message priority for this queue, in the range 0 (lowest) through 9 (highest).

If the **Priority** property is set to `Specified`, messages sent to this queue have the priority value specified by this property.

Data type Integer
Units Message priority level
Range 0 (lowest priority) through 9 (highest priority)

Expiry:

Whether the expiry timeout for this queue is defined by the application or the **Specified expiry** property, or whether messages on the queue expire (have an unlimited expiry timeout).

Data type Enum

Default APPLICATION_DEFINED

Range

Application defined
The expiry timeout for messages in this queue is defined by the application that put them onto the queue.

Specified
The expiry timeout for messages in this queue is defined by the **Specified expiry** property.If you select this option, you must define a time out on the **Specified expiry** property.

Unlimited
Messages in this queue have no expiry timeout, and those messages never expire.

Specified Expiry:

If the **Expiry timeout** property is set to `Specified`, specify the number of milliseconds (greater than 0) after which messages on this queue expire.

Data type Integer

Units Milliseconds

Range Greater than or equal to 0

- 0 indicates that messages never timeout.
- Other values are an integer number of milliseconds.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

V5 default messaging provider topic destination settings for application clients:

Use this panel to view or change the configuration properties of the selected topic destination for use with the internal product Java Message Service (JMS) provider.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **Version 5 Default Provider**. Right click **Topic Destinations** and click **New**. The following fields appear on the **General** tab.

A topic destination is used to configure the properties of a JMS topic for the associated JMS provider. A V5 default messaging provider topic has the following properties.

Name:

The name by which the topic is known for administrative purposes.

Data type String

Description:

A description of the topic, for administrative purposes within WebSphere Application Server.

Data type String

JNDI Name:

The application client run-time environment uses this field to retrieve configuration information.

Topic Name: The name of the topic as defined to the JMS provider.

Data type String

Persistence:

Whether all messages sent to the destination are persistent, nonpersistent, or have their persistence defined by the application.

Data type Enum
Default APPLICATION_DEFINED
Range

Application defined
Messages on the destination have their persistence defined by the application that put them onto the queue.

Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.

Persistent
Messages on the destination are persistent.

Nonpersistent
Messages on the destination are not persistent.

Priority:

Whether the message priority for this destination is defined by the application or the **Specified priority** property.

Data type Enum
Default APPLICATION_DEFINED
Range

Application defined
The priority of messages on this destination is defined by the application that put them onto the destination.

Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.

Specified
The priority of messages on this destination is defined by the **Specified priority** property.*If you select this option, you must define a priority on the **Specified priority** property.*

Specified Priority:

If the **Priority** property is set to `Specified`, specify the message priority for this queue, in the range 0 (lowest) through 9 (highest).

If the **Priority** property is set to `Specified`, messages sent to this queue have the priority value specified by this property.

Data type	Integer
Units	Message priority level
Range	0 (lowest priority) through 9 (highest priority)

Expiry:

Whether the expiry timeout for this queue is defined by the application or the **Specified expiry** property, or messages on the queue never expire (have an unlimited expiry timeout).

Data type	Enum
Default	APPLICATION_DEFINED
Range	**Application defined** The expiry timeout for messages on this queue is defined by the application that put them onto the queue. **Specified** The expiry timeout for messages on this queue is defined by the **Specified expiry** property. *If you select this option, you must define a timeout on the **Specified expiry** property.* **Unlimited** Messages on this queue have no expiry timeout, so those messages never expire.

Specified Expiry:

If the **Expiry timeout** property is set to `Specified`, type here the number of milliseconds (greater than 0) after which messages on this queue expire.

Data type	Integer
Units	Milliseconds
Range	Greater than or equal to 0 • 0 indicates that messages never time out. • Other values are an integer number of milliseconds.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

WebSphere MQ Provider queue connection factory settings for application clients:

Use this panel to view or change the configuration properties of the selected queue connection factory for use with the WebSphere MQ Java Message Service (JMS) provider. These configuration properties control how connections are created between the JMS provider and WebSphere MQ.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file and click **Messaging Providers** > **WebSphere MQ Provider**. Right click **Queue Connection Factories**, and click **New**. The following fields are displayed on the **General** tab.

Note:

- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the *Using Java* section of the WebSphere MQ information center.
- In WebSphere MQ, names can have a maximum of 48 characters, except for channels which have a maximum of 20 characters.

A queue connection factory for the JMS provider has the following properties.

Name:

The name by which this queue connection factory is known for administrative purposes within WebSphere Application Server. The name must be unique within the JMS connection factories across the WebSphere administrative domain.

Data type String

Description:

A description of this connection factory for administrative purposes within WebSphere Application Server.

Data type String
Default Null

JNDI Name:

The application client run time uses this field to retrieve configuration information.

User ID:

The user ID used, with the password property, for authentication if the calling application does not provide a user ID and password explicitly.

If you specify a value for the user ID property, you must also specify a value for the password property.

The connection factory user ID and password properties are used if the calling application does not provide a user ID and password explicitly; for example, if the calling application uses the method createQueueConnection(). The JMS client flows the user ID and password to the JMS server.

Data type String

Password:

The password used, with the user ID property, for authentication if the calling application does not provide a user ID and password explicitly.

If you specify a value for the user ID property, you must also specify a value for the password property.

Data type String
Default Null

Re-Enter Password:

Confirms the password.

Queue Manager:

The name of the WebSphere MQ queue manager for this connection factory.

Connections created by this factory connect to that queue manager.

Data type	String

Enter Hostname and Port Information:

This radio button is selected by default and, if selected, enables the host and port properties and disables the connection name list property.

Data type	Radio button
Default	Selected

Host:

The name of the host on which the WebSphere MQ queue manager runs for client connection only.

Data type	String
Default	Null
Range	A valid TCP/IP host name

Port:

The TCP/IP port number used for connection to the WebSphere MQ queue manager, for client connection only.

This port must be configured on the WebSphere MQ queue manager.

Data type	Integer
Default	Null
Range	A valid TCP/IP port number, configured on the WebSphere MQ queue manager.

Enter Connection Name List Information:

If selected, this radio button enables the connection name list property and disables the host and port name properties. Select this radio button if you want to connect to a multi-instance queue manager.

Data type	Radio button
Default	Cleared

Connection Name List:

A comma-separated list of host and port information which can be used to connect to a multi-instance queue manager.

The format of the list is:
`host[(port)],[host[(port)]]`

where port is optional and defaults to 1414 if it is not set. For example:
`hostname1,hostname2(1415)`

For further information about multi-instance queue managers, see the WebSphere MQ information center.

This property must only be used for connecting to a multi-instance queue manager. It must not be used for connecting to a list of distinct queue managers as that can result in transaction integrity issues.

Channel:

The name of the channel used for connection to the WebSphere MQ queue manager, for client connection only.

Data type	String
Default	Null
Range	1 through 20 ASCII characters

Transport type:

Specifies whether the WebSphere MQ client connection or JNDI bindings are used for connection to the WebSphere MQ queue manager. The external JMS provider controls the communication protocols between JMS clients and JMS servers. Tune the transport type when you are using non-ASF nonpersistent, nondurable, nontransactional messaging or when you want to satisfy security issues and the client is local to the queue manager node.

Data type	Enum
Units	Not applicable
Default	BINDINGS
Range	**BINDINGS** JNDI bindings are used to connect to the queue manager. BINDINGS is a shared memory protocol and can only be used when the queue manager is on the same node as the JMS client and poses security risks that must be addressed through the use of EJB roles. **CLIENT** WebSphere MQ client connection is used to connect to the queue manager. CLIENT is a typical TCP-based protocol. **DIRECT** For WebSphere MQ Event Broker using DIRECT mode. DIRECT is a lightweight sockets protocol used in nontransactional, nondurable, and nonpersistent Publish/Subscribe messaging. DIRECT only works for clients and message-driven beans using the non-ASF protocol. **QUEUED** QUEUED is a standard TCP protocol.

Recommended

Queue connection factory transport type

BINDINGS is faster by 30% or more, but it requires correctly set up EJB roles to guarantee security. If you have security concerns and need to use CLIENT then you should make appropriate use of SSL to secure the connection to the queue manager.

Topic connection factory transport type

DIRECT is the fastest type and must be used where possible. Use BINDINGS when you want to satisfy additional security tasks and the queue manager is local to the JMS client. QUEUED is the fallback for all other cases. WebSphere MQ 5.3 before CSD2 with the DIRECT setting can lose messages when used with message-driven beans and under load. This loss also happens with client-side applications unless the broker maxClientQueueSize is set to 0. You can set this value to 0 with the command:

```
#wempschangeproperties WAS_nodeName_server1
 -e default -o DynamicSubscriptionEngine -n
 maxClientQueueSize -v 0 -x executionGroupUUID
```

where executionGroupUUID can be found by starting the broker and looking in the Event Log/Applications for event 2201. This value is usually ffffffff-0000-0000-000000000000.

Note: The WebSphere MQ 5.3 JMS cannot be used within WebSphere Application Server Version 6.1 because WebSphere Application Server Version 6.1 has a Java 5 runtime. Therefore, cross-memory connections cannot be established with WebSphere MQ 5.3 queue managers. This can result in a performance degradation if you were previously using WebSphere MQ 5.3 and BINDINGS for your connections and move to CLIENT network connections in migrating to WebSphere Application Server Version 6.1.

Client ID:

The JMS client identifier used for connections to the WebSphere MQ queue manager.

Data type String

CCSID:

The coded character set identifier for use with the WebSphere MQ queue manager.

This coded character set identifier (CCSID) must be one of the CCSIDs supported by WebSphere MQ.

Data type String

For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *System Administration* and *Application Programming Reference* sections of the WebSphere MQ information center.

Message Retention:

Select this check box to specify that unwanted messages are to be left on the queue. Otherwise, unwanted messages are handled according to their disposition options.

Data type Enum
Units Not applicable
Default Cleared

Range

Selected
Unwanted messages are left on the queue.

Cleared
Unwanted messages are handled according to their disposition options.

Temporary model:

The name of the model definition used to create temporary connection factories if a connection factory does not already exist.

Data type String
Range 1 through 48 ASCII characters

Temporary queue prefix:

The prefix used for dynamic queue naming.

Data type String

Fail if quiesce:

Specifies whether applications return from a method call if the queue manager has entered a controlled failure.

Data type Check box
Default Selected

Local Server Address:

Specifies the local server address.

Data type String

Polling Interval:

Specifies the interval, in milliseconds, between scans of all receivers during asynchronous message delivery

Data type Integer
Units Milliseconds
Default 5000

Rescan interval:

Specifies the interval in milliseconds between which a topic is scanned to look for messages that have been added to a topic out of order.

This interval controls the scanning for messages that have been added to a topic out of order with respect to a WebSphere MQ browse cursor.

Data type Integer
Units Milliseconds

Default 5000

SSL cipher suite:

Specifies the cipher suite to use for SSL connection to WebSphere MQ.

Set this property to a valid cipher suite provided by your JSSE provider. The value must match the CipherSpec specified on the SVRCONN channel as the **Channel** property.

You must set this property, if you set the **SSL Peer Name** property.

SSL certificate store:

Specifies a list of zero or more Certificate Revocation List (CRL) servers used to check for SSL certificate revocation. If you specify a value for this property, you must use WebSphere MQ JVM at Java 2 version 1.4.

The value is a space-delimited list of entries of the form:
`ldap://hostname:[port]`

A single slash (/) follows this value. If *port* is omitted, the default LDAP port of 389 is assumed. At connect-time, the SSL certificate presented by the server is checked against the specified CRL servers. For more information about CRL security, see the information about "Working with Certificate Revocation Lists" in the *Security* section of the WebSphere MQ information center.

SSL peer name:

For SSL, a *distinguished name* skeleton that must match the name provided by the WebSphere MQ queue manager. The distinguished name is used to check the identifying certificate presented by the server at connection time.

If this property is not set, such certificate checking is performed.

The SSL peer name property is ignored if **SSL Cipher Suite** property is not specified.

This property is a list of attribute name and value pairs separated by commas or semicolons. For example:
`CN=QMGR.*, OU=IBM, OU=WEBSPHERE`

The example given checks the identifying certificate presented by the server at connect-time. For the connection to succeed, the certificate must have a Common Name beginning QMGR., and must have at least two Organizational Unit names, the first of which is IBM and the second WEBSPHERE. Checking is not case-sensitive.

For more details about distinguished names and their use with WebSphere MQ, see the information about "Distinguished Names" in the WebSphere MQ information center.

Connection pool:

Specifies an optional set of connection pool settings.

Connection pool properties are common to all J2C connectors.

The application server pools connections and sessions with the JMS provider to improve performance. This connection pooling is independent from any WebSphere MQ connection pooling. You must configure the connection and session pool properties appropriately for your applications, otherwise you might not get the connection and session behavior that you want.

Change the size of the connection pool if concurrent server-side access to the JMS resource exceeds the default value. The size of the connection pool is set on a per queue or topic basis.

Data type Check box
Default Selected

Client reconnect options:

Specifies whether a client mode connection reconnects automatically, or not, in the event of a communications or queue manager failure. This property is ignored unless the connection factory is being used in a thin or managed client environment.

Data type Drop-down list
Default DISABLED
Range

DISABLED
The client reconnection does not automatically occur.

ASDEF The value from the DefRecon attribute from the channels stanza of the client configuration file is used. If there is no DefRecon value specified then this setting has the same effect as a value of DISABLED.

RECONNECT
Reconnection occurs to any queue manager consistent with the value of the queue manager attribute, which might be a different queue manager from that to which the connection was originally connected.

QMGR Reconnection only occurs to the queue manager to which the connection was originally connected.

For more information about automatic client reconnection, see the WebSphere MQ information center.

Client reconnect timeout:

The maximum number of seconds that a client mode connection spends attempting to automatically reconnect to a queue manager after a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment. Whether this parameter is used or not depends on the value of the client reconnect options parameter.

Data type Integer
Units Seconds
Default 1800
Range A value greater than zero and up to 2147483647

For more information about automatic client reconnection, see the WebSphere MQ information center.

WebSphere MQ Provider topic connection factory settings for application clients:

Use this panel to view or change the configuration properties of the selected topic connection factory for use with the WebSphere MQ Java Message Service (JMS) provider. These configuration properties control how connections are created between the JMS provider and WebSphere MQ.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **WebSphere MQ Provider**. Right-click **Topic Connection Factories** and click **New**.

Note:

- The property values that you specify must match the values that you specified when configuring WebSphere MQ product JMS resources. For more information about configuring WebSphere MQ JMS resources, see the *Using Java* section of the WebSphere MQ information center.
- In WebSphere MQ, names can have a maximum of 48 characters, except for channels which have a maximum of 20 characters.

MA0C broker: When creating a WebSphere Application Server Version 6 topic connection factory for the MA0C broker, consider the following attribute values:

BrokerControlQueue
This value is fixed at SYSTEM.BROKER.CONTROL.QUEUE for the MA0C broker and is the queue the broker reads from.

BrokerVersion
Set this value to BASIC for the MA0C broker.

ClientID
Set this value to whatever you like for the MA0C broker (the value is string and is merely an identifier for your client application).

XA Enabled
Set this value to TRUE or FALSE for the MA0C broker (the setting you use is a performance enhancement flag - you probably want to set this value to 'true' most of the time).

BrokerMessage Selection
This value is fixed at CLIENT for the MA0C broker because the broker relies on client side message selection.

Direct Broker Authorization Type
This value is not required by the MA0C broker.

A topic connection factory for the WebSphere MQ JMS provider has the following properties.

Name:

The name by which this topic connection factory is known for administrative purposes within WebSphere Application Server. The name must be unique within the JMS provider.

Data type String

Description:

A description of this topic connection factory for administrative purposes within WebSphere Application Server.

Data type String

JNDI Name:

The Java Naming and Directory Interface (JNDI) name that is used to bind the topic connection factory into the application server name space.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String
Units En_US ASCII characters
Range 1 through 45 ASCII characters

User ID:

The user ID used, with the password property, for authentication if the calling application does not provide a user ID and password explicitly.

If you specify a value for the user ID property, you must also specify a value for the password property.

The connection factory user ID and password properties are used if the calling application does not provide a user ID and password explicitly, for example, if the calling application uses the method createTopicConnection(). The JMS client flows the user ID and password to the JMS server.

Data type String

Password:

The password used, with the user ID property, for authentication if the calling application does not provide a user ID and password explicitly.

If you specify a value for the user ID property, you must also specify a value for the password property.

Data type String

Re-Enter Password:

Confirms the password.

Queue Manager:

The name of the WebSphere MQ queue manager for this connection factory. Connections created by this connection factory connect to this queue manager.

Data type String

Enter Hostname and Port Information:

This radio button is selected by default and, if selected, enables the host and port properties and disables the connection name list property.

Data type Radio button
Default Selected

Host:

The name of the host on which the WebSphere MQ queue manager runs for client connections only.

Data type	String
Range	A valid TCP/IP host name

Port:

The TCP/IP port number used for connection to the WebSphere MQ queue manager, for client connection only.

This port must be configured on the WebSphere MQ queue manager.

Data type	Integer
Range	A valid TCP/IP port number, configured on the WebSphere MQ queue manager.

Enter Connection Name List Information:

If selected, this radio button enables the connection name list property and disables the host and port name properties. Select this radio button if you want to connect to a multi-instance queue manager.

Data type	Radio button
Default	Cleared

Connection Name List:

A comma-separated list of host and port information which can be used to connect to a multi-instance queue manager.

The format of the list is:

```
host[(port)],[host[(port)]]
```

where port is optional and defaults to 1414 if it is not set. For example:

```
hostname1,hostname2(1415)
```

For further information about multi-instance queue managers, see the WebSphere MQ information center.

This property must only be used for connecting to a multi-instance queue manager. It must not be used for connecting to a list of distinct queue managers as that can result in transaction integrity issues.

Channel:

The name of the channel used for client connections to the WebSphere MQ queue manager, for client connection only.

Data type	String
Range	1 through 20 ASCII characters

Transport Type:

Whether WebSphere MQ client connection or JNDI bindings are used for connection to the WebSphere MQ queue manager.

Data type Enum
Default BINDINGS
Range
CLIENT
WebSphere MQ client connection is used to connect to the WebSphere MQ queue manager.
BINDINGS
JNDI bindings are used to connect to the WebSphere MQ queue manager.

Client ID:

The JMS client identifier used for connections to the WebSphere MQ queue manager.

Data type String

CCSID:

The coded character set identifier to use with the WebSphere MQ queue manager.

This coded character set identifier (CCSID) must be one of the CCSIDs that WebSphere MQ supports. See the properties for the topic destination for more details.

Data type String
Units Integer
Range 1 through 65535

Broker Control Queue:

The name of the broker control queue to which all command messages (except publications and requests to delete publications) are sent.

Data type String
Units En_US ASCII characters
Range 1 through 48 ASCII characters

Broker Queue Manager:

The name of the WebSphere MQ queue manager that provides the Publisher and Subscriber message broker.

Data type String
Units En_US ASCII characters
Range 1 through 48 ASCII characters

Broker Publish Queue:

The name of the broker input queue that receives all publication messages for the default stream.

The name of the broker's input queue (stream queue) that receives all publication messages for the default stream. Applications can also send requests to delete publications on the default stream to this queue.

Data type String
Units En_US ASCII characters
Range 1 through 48 ASCII characters

Broker Subscribe Queue:

The name of the broker queue from which nondurable subscription messages are retrieved.

The name of the broker queue from which nondurable subscription messages are retrieved. The subscriber specifies the name of the queue when it registers a subscription.

Data type String
Units En_US ASCII characters
Range 1 through 48 ASCII characters

Broker CCSubQ:

The name of the broker queue from which nondurable subscription messages are retrieved for a ConnectionConsumer request. This property applies only for use of the web container.

Data type String
Units En_US ASCII characters
Range 1 through 48 ASCII characters

Broker Version:

Whether the message broker is provided by the WebSphere MQ MA0C SupportPac or newer versions of WebSphere family message broker products.

Data type Enum
Default Advanced
Range

Advanced
The message broker is provided by newer versions of WebSphere family message broker products (WebSphere MQ Integrator and WebSphere MQ Publish and Subscribe).

Basic
The message broker is provided by the WebSphere MQ MA0C SupportPac (WebSphere MQ - Publish and Subscribe).

Cleanup level:

The level of cleanup provided by the publish or subscribe cleanup utility.

Data type Enum
Default SAFE
Range

ASPROP

NONE

STRONG

Cleanup interval:

The interval, in milliseconds, between background executions of the publish/subscribe cleanup utility.

Data type Integer
Units Milliseconds
Default 6000

Message selection:

Where broker message selection is performed.

Data type Enum
Default BROKER
Range

BROKER
Message selection is performed at the broker location.

Message CLIENT
Message selection is performed at the client location.

Publish acknowledge interval:

The interval, in number of messages, between publish requests that require acknowledgment from the broker.

Data type Integer
Default 25

Sparse subscriptions:

Enables sparse subscriptions.

Data type Check box
Default Cleared

Status refresh interval:

The interval, in milliseconds, between transactions to refresh the publish or subscribe status.

Data type Integer
Default 6000

Subscription store:

Where WebSphere MQ stores data relating to active JMS subscriptions.

Data type Enum
Default MIGRATE
Range

MIGRATE

QUEUE

BROKER

Multicast:

Whether this connection factory uses multicast transport.

Data type Enum
Default NOT USED
Range

NOT USED
This connection factory does not use multicast transport.

ENABLED
This connection factory always uses multicast transport.

ENABLED_IF_AVAILABLE
This connection factory uses multicast transport.

ENABLED_RELIABLE
This connection factory uses reliable multicast transport.

ENABLED_RELIABLE_IF_AVAILABLE
This connection factory uses reliable multicast transport if available.

Direct authentication:

Whether to use direct broker authorization.

Data type Enum
Default NONE
Range

NONE Direct broker authorization is not used.

PASSWORD
Direct broker authorization is authenticated with a password.

CERTIFICATE
Direct broker authorization is authenticated with a certificate.

Proxy Host Name:

The host name of a proxy to be used for communication with WebSphere MQ.

Data type String

Proxy Port:

The port number of a proxy to be used for communication with WebSphere MQ.

Data type Integer
Default 0

Fail if quiesce:

Whether applications return from a method call if the queue manager has entered a controlled failure.

Data type	Check box
Default	Selected

Local Server Address:

The local server address.

Data type	String

Polling Interval:

The interval, in milliseconds, between scans of all receivers during asynchronous message delivery.

Data type	Integer
Units	Milliseconds
Default	5000

Rescan interval:

The interval in milliseconds between which a topic is scanned to look for messages that have been added to a topic out of order.

The rescan interval controls the scanning for messages that have been added to a topic out of order with respect to a WebSphere MQ browse cursor.

Data type	Integer
Units	Milliseconds
Default	5000

SSL cipher suite:

The cipher suite to use for SSL connection to WebSphere MQ.

Set this property to a valid cipher suite provided by your JSSE provider. The value must match the CipherSpec specified on the SVRCONN channel as the **Channel** property.

You must set this property, if you set the **SSL Peer Name** property.

SSL certificate store:

A list of zero or more Certificate Revocation List (CRL) servers that are used to check for SSL certificate revocation. If you specify a value for this property, you must use WebSphere MQ JVM at Java 2 version 1.4.

The value is a space-delimited list of entries of the form:
`ldap://hostname:[port]`

A single slash (/) follows this value. If *port* is omitted, the default LDAP port of 389 is assumed. At connect-time, the SSL certificate presented by the server is checked against the specified CRL servers. For more information about CRL security, see the information about “Working with Certificate Revocation Lists” in the *Security* section of the WebSphere MQ information center.

SSL peer name:

For SSL, a *distinguished name* skeleton that must match the name provided by the WebSphere MQ queue manager. The distinguished name is used to check the identifying certificate presented by the server at connection time.

If this property is not set, such certificate checking is performed.

The SSL peer name property is ignored if **SSL Cipher Suite** property is not specified.

This property is a list of attribute name and value pairs separated by commas or semicolons. For example:

```
CN=QMGR.*, OU=IBM, OU=WEBSPHERE
```

The example given checks the identifying certificate presented by the server at connect-time. For the connection to succeed, the certificate must have a Common Name beginning QMGR., and must have at least two Organizational Unit names, the first of which is IBM and the second WEBSPHERE. Checking is not case-sensitive.

For more details about distinguished names and their use with WebSphere MQ, see the information about "Distinguished Names" in the Security section of the WebSphere MQ information center.

Connection pool:

An optional set of connection pool settings.

Connection pool properties are common to all J2C connectors.

The application server pools connections and sessions with the JMS provider to improve performance. This connection pooling is independent from any WebSphere MQ connection pooling. You must configure the connection and session pool properties appropriately for your applications, otherwise you might not get the connection and session behavior that you want.

Change the size of the connection pool if concurrent server-side access to the JMS resource exceeds the default value. The size of the connection pool is set on a per queue or topic basis.

Data type	Check box
Default	Selected

Client reconnect options:

Specifies whether a client mode connection reconnects automatically, or not, in the event of a communications or queue manager failure. This property is ignored unless the connection factory is being used in a thin or managed client environment.

Data type	Drop-down list
Default	DISABLED

Range

DISABLED
The client reconnection does not automatically occur.

ASDEF The value from the DefRecon attribute from the channels stanza of the client configuration file is used. If there is no DefRecon value specified then this setting has the same effect as a value of DISABLED.

RECONNECT
Reconnection occurs to any queue manager consistent with the value of the queue manager attribute, which might be a different queue manager from that to which the connection was originally connected.

QMGR Reconnection only occurs to the queue manager to which the connection was originally connected.

For more information about automatic client reconnection, see the WebSphere MQ information center.

Client reconnect timeout:

The maximum number of seconds that a client mode connection spends attempting to automatically reconnect to a queue manager after a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment. Whether this parameter is used or not depends on the value of the client reconnect options parameter.

Data type	Integer
Units	Seconds
Default	1800
Range	A value greater than zero and up to 2147483647

For more information about automatic client reconnection, see the WebSphere MQ information center.

WebSphere MQ Provider queue destination settings for application clients:

Use this panel to view or change the configuration properties of the selected queue destination for use with the WebSphere MQ product Java Message Service (JMS) provider.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file and click **Messaging Providers** > **WebSphere MQ Provider**. Right-click **Queue Destinations** and click **New**. The following fields are displayed on the **General** tab.

Note:
- The property values that you specify must match the values that you specified when configuring JMS resources for WebSphere MQ. For more information about configuring JMS resources for WebSphere MQ, see *Using Java* in the WebSphere MQ information center.
- In WebSphere MQ, names can have a maximum of 48 characters.

A queue for use with the WebSphere MQ product JMS provider has the following properties.

Name:

The name by which the queue is known for administrative purposes within WebSphere Application Server.

Data type String

Description:

A description of the queue, for administrative purposes within WebSphere Application Server.

Data type String

JNDI Name:

The application client runtime environment uses this field to retrieve configuration information.

Persistence:

Whether all messages sent to the destination are persistent, nonpersistent or have their persistence defined by the application.

Data type Enum
Default APPLICATION_DEFINED
Range

Application defined
Messages on the destination have their persistence defined by the application that put them onto the queue.

Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.

Persistent
Messages on the destination are persistent.

Nonpersistent
Messages on the destination are not persistent.

Priority:

Whether the message priority for this destination is defined by the application or the **Specified priority** property.

Data type Enum
Units Not applicable
Default APPLICATION_DEFINED
Range

Application defined
The priority of messages on this destination is defined by the application that put them onto the destination.

Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.

Specified
The priority of messages on this destination is defined by the **Specified priority** property. *If you select this option, you must define a priority on the **Specified priority** property.*

Specified Priority:

If the **Priority** property is set to `Specified`, specify the message priority for this queue, in the range 0 (lowest) through 9 (highest).

Data type Integer
Units Message priority level
Range 0 (lowest priority) through 9 (highest priority)

Expiry:

Whether the expiry timeout value for this queue is defined by the application or the by **Specified expiry** property or whether messages on the queue never expire (have an unlimited expiry time out).

Data type Enum
Units Not applicable
Default APPLICATION_DEFINED
Range
Application defined
The expiry timeout for messages on this queue is defined by the application that put them onto the queue.
Specified
The expiry timeout for messages on this queue is defined by the **Specified expiry** property. If you select this option, you must define a timeout on the **Specified expiry** property.
Unlimited
Messages on this queue have no expiry timeout and those messages never expire.

Specified Expiry:

If the **Expiry timeout** property is set to `Specified`, type here the number of milliseconds (greater than 0) after which messages on this queue expire.

Data type Integer
Units Milliseconds
Range Greater than or equal to 0
- 0 indicates that messages never time out
- Other values are an integer number of milliseconds

Base Queue Name:

The name of the queue to which messages are sent, on the queue manager specified by the **Base queue manager name** property.

Data type String

Base Queue Manager Name:

The name of the WebSphere MQ queue manager to which messages are sent.

This queue manager provides the queue specified by the **Base queue name** property.

Data type String
Units En_US ASCII characters
Range A valid WebSphere MQ Queue Manager name, as 1 through 48 ASCII characters

CCSID:

The coded character set identifier to use with the WebSphere MQ queue manager.

This coded character set identifier (CCSID) must be one of the CCSIDs supported by WebSphere MQ queue manager. See the WebSphere MQ messaging provider queue and topic advanced properties settings for more details.

Data type String

Integer encoding:

If native encoding is not enabled, select whether integer encoding is normal or reversed.

Data type Enum
Default NORMAL
Range
NORMAL
Normal integer encoding is used.
REVERSED
Reversed integer encoding is used.

For more information about encoding properties, see *Using Java* in the WebSphere MQ information center.

Decimal encoding:

If native encoding is not enabled, select whether decimal encoding is normal or reversed.

Data type Enum
Default NORMAL
Range
NORMAL
Normal decimal encoding is used.
REVERSED
Reversed decimal encoding is used.

For more information about encoding properties, see *Using Java* in the WebSphere MQ information center.

Floating point encoding:

If native encoding is not enabled, select the type of floating point encoding.

Data type Enum
Default IEEENORMAL
Range
IEEENORMAL
IEEE normal floating point encoding is used.
IEEEREVERSED
IEEE reversed floating point encoding is used.
S390 S390 floating point encoding is used.

For more information about encoding properties, see *Using Java* in the WebSphere MQ information center.

Native encoding:

Indicates that the queue destination uses native encoding (appropriate encoding values for the Java platform) when you select this check box.

Data type Enum
Default Cleared
Range **Cleared**
Native encoding is not used, so specify the following properties for integer, decimal and floating point encoding.
Selected
Native encoding is used (to provide appropriate encoding values for the Java platform).

For more information about encoding properties, see *Using Java* in the WebSphere MQ information center.

Target client:

Whether the receiving application is JMS compliant or is a traditional WebSphere MQ application.

Data type Enum
Default WebSphere MQ
Range **WebSphere MQ**
The target is a traditional WebSphere MQ application that does not support JMS.
JMS The target application supports JMS.

Message body:

Specifies whether an application processes the RFH version 2 header of a WebSphere MQ message as part of the JMS message body.

Data type Drop-down list
Default UNSPECIFIED
Range **UNSPECIFIED**
When sending messages, the WebSphere MQ messaging provider does or does not generate and include an RFH version 2 header, depending on the value of the Append RFH version 2 headers to messages sent to this destination property. When receiving messages, the WebSphere MQ messaging provider acts as if the value is set to JMS.
JMS When sending messages, the WebSphere MQ messaging provider automatically generates an RFH version 2 header and includes it in the WebSphere MQ message. When receiving messages, the WebSphere MQ messaging provider sets the JMS message properties according to values in the RFH version 2 header (if these value are present); it does not present the RFH version 2 header as part of the JMS message body.
MQ When sending messages, the WebSphere MQ messaging provider does not generate an RFH version 2 header. When receiving messages, the WebSphere MQ messaging provider presents the RFH version 2 header as part of the JMS message body.

ReplyTo destination style:

Specifies the format of the JMSReplyTo field.

Data type Drop-down list
Default DEFAULT

Range

DEFAULT
The default value is equivalent to the information in the RFH version 2 header.

MQMD Use the value supplied in the MQMD. This populates the reply to queue manager field with the value from the MQMD, equivalent to the default behaviour of WebSphere MQ Version 6.0.2.4 and 6.0.2.5.

RFH2 Use the value supplied in the RFH version 2 header. If the sending application set a JMSReplyTo value, then that value is used.

MQMD read enabled:

Specifies whether an application can read the values of MQMD fields from JMS messages that have been sent or received using the WebSphere MQ messaging provider.

Data type Check box

Default Cleared

Range

Cleared
Applications cannot read the values of the MQMD fields.

Selected
Applications can read the values of the MQMD fields.

MQMD write enabled:

Specifies whether an application can write the values of MQMD fields to JMS messages that will be sent or received using the WebSphere MQ messaging provider.

Data type Check box

Default Cleared

Range

Cleared
Applications cannot write the values of the MQMD fields.

Selected
Applications can write the values of the MQMD fields.

MQMD message context:

Defines the message context options specified when sending messages to a destination.

Data type Drop-down list

Default DEFAULT

Range

DEFAULT
The MQOPEN API call and the MQPMO structure specify no explicit message context options.

SET_IDENTITY_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_IDENTITY_CONTEXT, and the MQPMO structure specifies MQPMO_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_ALL_CONTEXT, and the MQPMO structure specifies MQPMO_SET_ALL_CONTEXT.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

WebSphere MQ Provider topic destination settings for application clients:

Use this panel to view or change the configuration properties of the selected topic destination for use with the WebSphere MQ product Java Message Service (JMS) provider.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > **WebSphere MQ Provider**. Right click **Topic Destinations**, and click **New**. The following fields are displayed on the **General** tab.

Note:

- The property values that you specify must match the values that you specified when configuring JMS resources for WebSphere MQ. For more information about configuring JMS resources for WebSphere MQ, see *Using Java* in the WebSphere MQ information center.
- In WebSphere MQ, names can have a maximum of 48 characters.

A topic destination is used to configure the properties of a JMS topic for the associated JMS provider. A topic for use with the WebSphere MQ product JMS provider has the following properties.

Name:

The name by which the topic is known for administrative purposes within WebSphere Application Server.

Data type String

Description:

A description of the topic for administrative purposes within WebSphere Application Server.

Data type String

JNDI Name:

The application client runtime environment uses this field to retrieve configuration information.

Persistence:

Whether all messages sent to the destination are persistent, nonpersistent, or have their persistence defined by the application.

Data type Enum
Default APPLICATION_DEFINED
Range
Application defined
Messages on the destination have their persistence defined by the application that put them in the queue.
Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.
Persistent
Messages on the destination are persistent.
Nonpersistent
Messages on the destination are not persistent.

Priority:

Whether the message priority for this destination is defined by the application or the **Specified priority** property.

Data type Enum
Default APPLICATION_DEFINED
Range
Application defined
The priority of messages on this destination is defined by the application that put them in the destination.
Queue defined
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.
Specified
The priority of messages on this destination is defined by the **Specified priority** property. If you select this option, you must define a priority for the **Specified priority** property.

Specified Priority:

If the **Priority** property is set to `Specified`, specify the message priority for this queue, in the range 0 (lowest) through 9 (highest).

If the **Priority** property is set to `Specified`, messages sent to this queue have the priority value specified by this property.

Data type Integer
Units Message priority level
Range 0 (lowest priority) through 9 (highest priority)

Expiry:

Whether the expiry timeout for this queue is defined by the application or by the **Specified expiry** property, or whether messages on the queue never expire (have an unlimited expiry timeout).

Data type Enum
Default APPLICATION_DEFINED
Range
Application defined
The expiry timeout for messages on this queue is defined by the application that put them in the queue.
Specified
The expiry timeout for messages in this queue is defined by the **Specified expiry** property. If you select this option, you must define a timeout value for the **Specified expiry** property.
Unlimited
Messages on this queue have no expiry timeout, and these messages never expire.

Specified Expiry:

If the **Expiry timeout** property is set to `Specified`, type the number of milliseconds (greater than 0) after which messages on this queue expire.

Data type Integer

Units Milliseconds
Range Greater than or equal to 0
- 0 indicates that messages never time out.
- Other values are an integer number of milliseconds.

Base Topic Name:

The name of the topic to which messages are sent.

Data type String

CCSID:

The coded character set identifier to use with the WebSphere MQ queue manager.

This coded character set identifier (CCSID) must be one of the CCSIDs that WebSphere MQ supports.

Data type String
Units Integer
Range 1 through 65535

Integer encoding:

If native encoding is not enabled, select whether integer encoding is normal or reversed.

Data type Enum
Default NORMAL
Range
NORMAL
Normal integer encoding is used.
REVERSED
Reversed integer encoding is used.

For more information about encoding properties, see *Using Java* in the WebSphere MQ information center.

Decimal encoding:

If native encoding is not enabled, select whether decimal encoding is normal or reversed.

Data type Enum
Default NORMAL
Range
NORMAL
Normal decimal encoding is used.
REVERSED
Reversed decimal encoding is used.

For more information about encoding properties, see *Using Java* in the information WebSphere MQ center.

Floating point encoding:

If native encoding is not enabled, select the type of floating point encoding.

Data type Enum

Default IEEENORMAL

Range

IEEENORMAL
IEEE normal floating point encoding is used.

IEEEREVERSED
IEEE reversed floating point encoding is used.

S390 S/390® floating point encoding is used.

For more information about encoding properties, see *Using Java* in the WebSphere MQ information center.

Native encoding:

Indicates that the queue destination uses native encoding (appropriate encoding values for the Java platform) when you select this check box.

Data type Enum

Default Cleared

Range

Cleared
Native encoding is not used, so specify the previous properties for integer, decimal and floating point encoding.

Selected
Native encoding is used (to provide appropriate encoding values for the Java platform).

For more information about encoding properties, see *Using Java* in the WebSphere MQ information center.

BrokerDurSubQueue:

The name of the broker queue from which durable subscription messages are retrieved.

The subscriber specifies the name of the queue when it registers a subscription.

Data type String

Units En_US ASCII characters

Range 1 through 48 ASCII characters

BrokerCCDurSubQueue:

The name of the broker queue from which durable subscription messages are retrieved for a ConnectionConsumer. This property applies only for use of the web container.

Data type String

Units En_US ASCII characters

Range 1 through 48 ASCII characters

Target Client:

Whether the receiving application is JMS compliant or is a traditional WebSphere MQ application.

Data type Enum

Default WebSphere MQ

Range **WebSphere MQ**
The target is a traditional WebSphere MQ application that does not support JMS.
JMS The target application supports JMS.

Message body:

Specifies whether an application processes the RFH version 2 header of a WebSphere MQ message as part of the JMS message body.

Data type Drop-down list
Default UNSPECIFIED
Range **UNSPECIFIED**
When sending messages, the WebSphere MQ messaging provider does or does not generate and include an RFH version 2 header, depending on the value of the Append RFH version 2 headers to messages sent to this destination property. When receiving messages, the WebSphere MQ messaging provider acts as if the value is set to JMS.
JMS When sending messages, the WebSphere MQ messaging provider automatically generates an RFH version 2 header and includes it in the WebSphere MQ message. When receiving messages, the WebSphere MQ messaging provider sets the JMS message properties according to values in the RFH version 2 header (if these value are present); it does not present the RFH version 2 header as part of the JMS message body.
MQ When sending messages, the WebSphere MQ messaging provider does not generate an RFH version 2 header. When receiving messages, the WebSphere MQ messaging provider presents the RFH version 2 header as part of the JMS message body.

ReplyTo destination style:

Specifies the format of the JMSReplyTo field.

Data type Drop-down list
Default DEFAULT
Range **DEFAULT**
The default value is equivalent to the information in the RFH version 2 header.
MQMD Use the value supplied in the MQMD. This populates the reply to queue manager field with the value from the MQMD, equivalent to the default behaviour of WebSphere MQ Version 6.0.2.4 and 6.0.2.5.
RFH2 Use the value supplied in the RFH version 2 header. If the sending application set a JMSReplyTo value, then that value is used.

Multicast:

Whether this connection factory uses multicast transport.

Data type Enum
Default AS_CF

Range

AS_CF This connection factory uses multicast transport.

DISABLED
This connection factory does not use multicast transport.

NOT_RELIABLE
This connection factory always uses multicast transport.

RELIABLE
This connection factory uses multicast transport when the topic destination is not reliable.

ENABLED
This connection factory uses reliable multicast transport.

MQMD read enabled:

Specifies whether an application can read the values of MQMD fields from JMS messages that have been sent or received using the WebSphere MQ messaging provider.

Data type Check box
Default Cleared
Range

Cleared
Applications cannot read the values of the MQMD fields.

Selected
Applications can read the values of the MQMD fields.

MQMD write enabled:

Specifies whether an application can write the values of MQMD fields to JMS messages that will be sent or received using the WebSphere MQ messaging provider.

Data type Check box
Default Cleared
Range

Cleared
Applications cannot write the values of the MQMD fields.

Selected
Applications can write the values of the MQMD fields.

MQMD message context:

Defines the message context options specified when sending messages to a destination.

Data type Drop-down list
Default DEFAULT
Range

DEFAULT
The MQOPEN API call and the MQPMO structure specify no explicit message context options.

SET_IDENTITY_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_IDENTITY_CONTEXT, and the MQPMO structure specifies MQPMO_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_ALL_CONTEXT, and the MQPMO structure specifies MQPMO_SET_ALL_CONTEXT.

Generic JMS connection factory settings for application clients:

Use this panel to view or change the configuration properties of the selected Java Message Service (JMS) connection factory for use with the associated JMS provider. These configuration properties control how connections are created between the JMS provider and the messaging system that it uses.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > *new_JMS_Provider_instance*. Right-click **Connection Factories**, and click **New**. The following fields are displayed on the **General** tab.

A Java Message Service (JMS) connection factory creates connections to JMS destinations. The JMS connection factory is created by the associated JMS provider. A JMS connection factory for a generic JMS provider (other than the internal default messaging provider or WebSphere MQ as a JMS provider) has the following properties:

Name:

The name by which this JMS connection factory is known for administrative purposes within IBM WebSphere Application Server. The name must be unique within the associated JMS provider.

Data type String

Description:

A description of this connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String
Default Null

JNDI Name:

The application client run time uses this field to retrieve configuration information.

User ID:

Indicates the user ID used with the **Password** property, for authentication if the calling application does not provide a `userid` and `password` explicitly.

If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.

The connection factory **User ID** and **Password** properties are used if the calling application does not provide a userid and password explicitly; for example, if the calling application uses the method createQueueConnection(). The JMS client flows the `userid` and `password` to the JMS server.

Data type String

Password:

The password used with the **User ID** property for authentication if the calling application does not provide a `userid` and `password` explicitly.

If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.

Data type String
Default Null

Re-Enter Password:

Confirms the password entered in the **Password** field.

External JNDI Name:

The JNDI name that is used to bind the queue into the application server name space.

As a convention, use the fully qualified JNDI name, for example, `jms/`*`Name`*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI API by the platform.

Data type String

Connection Type:

Whether this JMS destination is a queue (for point-to-point) or topic (for publication or subscription).

Select one of the following options:
Queue
A JMS queue destination for point-to-point messaging.
Topic A JMS topic destination for publish subscribe messaging.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Generic JMS destination settings for application clients:

Use this panel to view or change the configuration properties of the selected JMS destination for use with the associated JMS provider.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Messaging Providers** > *new JMS Provider instance*. Right-click **Destinations**, and click **New**. The following fields are displayed on the **General** tab.

A JMS destination is used to configure the properties of a JMS destination for the associated generic JMS provider. Connections to the JMS destination are created by the associated JMS connection factory. A JMS destination for use with a generic JMS provider (not the default messaging provider or WebSphere MQ as a JMS provider) has the following properties.

Name:

The name by which the queue is known for administrative purposes within WebSphere Application Server.

Data type String

Description:

A description of the queue, for administrative purposes.

JNDI Name:

The JNDI name of the actual (physical) name of the JMS destination bound into JNDI.

External JNDI Name:

The JNDI name that is used to bind the queue into the application server name space.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Destination Type:

Whether this JMS destination is a queue (for point-to-point) or topic (for publishing or subscribing).

Select one of the following options:
Queue
A JMS queue destination for point-to-point messaging.
Topic A JMS topic destination for pub/sub messaging.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Example: Configuring JMS provider, JMS connection factory and JMS destination settings for application clients:

You can configure JMS Provider, JMS Connection Factory and JMS Destination settings. This topic provides the required fields, special cases, and an example.

The purpose of this article is to help you to configure JMS Provider, JMS Connection Factory and JMS Destination settings.
- Required fields include:
  - JMS Provider Properties page: name, and at least one protocol provider
  - JMS Connection Factory Properties page: name, jndiName, destination type
  - JMS Destination Properties page: name, jndiName, destination type

- Special cases:
  - The destination type must be `QUEUE`, or `TOPIC`.
- Example:

```
<resources.jms:JMSProvider xmi:id="JMSProvider_3" name="genericJMSProvider:name"
description="genericJMSProvider:description"
externalInitialContextFactory="genericJMSProvider:contextFactoryClass"
externalProviderURL="genericJMSProvider:providerUrl">
<classpath>genericJMSProvider:classpath</classpath>
<factories xmi:type="resources.jms:GenericJMSDestination"
xmi:id="GenericJMSDestination_1" name="jmsDestination:name"
jndiName="jmsDestination:jndiName" description="jmsDestination:description"
externalJNDIName="jmsDestination:externalJndiName" type="QUEUE">
<propertySet xmi:id="J2EEResourcePropertySet_15">
<resourceProperties xmi:id="J2EEResourceProperty_17" name="jmsDestination:customName"
value="jmsDestination:customValue"/>
</propertySet>
</factories>
<factories xmi:type="resources.jms:GenericJMSConnectionFactory"
xmi:id="GenericJMSConnectionFactory_1" name="jmsCF:name" jndiName="jmsCF:jndiName"
description="jmsCF:description" userID="jmsCF:user" password="{xor}NTIsHBllMT4yOg=="
externalJNDIName="jmsCF:externalJndiName" type="QUEUE">
<propertySet xmi:id="J2EEResourcePropertySet_16">
<resourceProperties xmi:id="J2EEResourceProperty_18" name="jmsCF:customName"
value="jmsCF:customValue"/>
</propertySet>
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_17">
<resourceProperties xmi:id="J2EEResourceProperty_19"
name="genericJMSProvider:customName" value="genericJMSProvider:customValue"/>
</propertySet>
</resources.jms:JMSProvider>
```

Configuring new JMS connection factories for application clients:

Use this task to create a new Java Message Service (JMS) connection factory configuration for your application client.

Procedure

1. Click the JMS provider for which you want to create a connection factory in the tree. Complete one of the following actions:
   - Configure a new JMS provider.
   - Click an existing JMS provider.
2. Expand the JMS provider to view its Connection Factories folder.
3. Click the connection factory folder, and complete one of the following actions:
   - Right-click the folder and select**New**.
   - Click **Edit > New** on the menu bar.
4. Configure the JMS connection factory properties in the displayed fields.
5. Click **OK** when you finish.
6. Click **File > Save** on the menu bar to save your changes.

Configuring new JMS destinations for application clients:

Use this task to create a new Java Message Service (JMS) destination configuration for your application client.

Procedure

1. Click the JMS provider in the tree for which you want to create a destination. Complete one of the following actions:
   - Configure a new JMS provider.

- Click an existing JMS provider.

2. Expand the JMS provider to view its Destinations folder.
3. Click the provider folder, and complete one of the following actions:
   - Right-click the folder and select **New**.
   - Click **Edit > New** on the menu bar.
4. Configure the JMS destination properties in the displayed fields.
5. Click **OK** when you finish.
6. Click **File > Save** on the menu bar to save your changes.

Configuring new resource environment providers for application clients:

You can create new resource environment provider configurations for your application client using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

During this task, you create new resource environment provider configurations for your application client.

About this task

To configure a new resource environment provider, perform the following steps:

Procedure

1. Start the Application Configuration Resource Tool and open the EAR file for which you want to configure the new Java Message Service (JMS) provider. The EAR file contents display in a tree view.
2. Select from the tree the JAR file in which you want to configure the new JMS provider.
3. Expand the JAR file to view its contents.
4. Click the **Resource Environment Providers** folder. Take one of the following actions:
   - Right-click the provider folder, and click **New**.
   - Click **Edit > New** on the menu bar.
5. Configure the JMS provider properties in the displayed fields.
6. Click **OK** when you finish.
7. Click **File > Save** on the menu bar to save your changes.

Resource environment provider settings for application clients:

Use this page to specify resource environment entry properties.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected Java Archive (JAR) file. Right-click **Resource Environment Providers**, and click **New**. The following fields are displayed on the **General** tab:

Name:

Specifies the administrative name for the resource environment provider.

Description:

Specifies a description of the resource environment provider for your administrative records.

Class Path:

Specifies the path to the JAR file that contains the implementation classes for the resource environment provider.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Configuring new resource environment entries for application clients:

You can create new resource environment entries for your client application using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

During this task, you create new resource environment entries for your client application.

About this task

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT).
2. Open the EAR file for which you want to configure the new resource environment entry. The EAR file contents are in the displayed tree view.
3. Click the desired resource environment provider, and complete the following action to configure new providers:
   - Configure a new resource environment provider.
4. Expand the resource environment provider to view the Resource Environment Entries folder.
5. Click the resource environment entries folder, and complete one of the following actions:
   - Right-click the folder and select **New**.
   - Click **Edit > New** on the menu bar.
6. Configure the resource environment entry properties in the displayed fields.
7. Click **OK**.
8. Click **File > Save** on the menu bar to save your changes.

Resource environment entry settings for application clients:

Use this page to specify resource environment entry properties.

To view this Application Client Resource Configuration Tool (ACRCT) page, click **File** > **Open**. After you browse for an EAR file, click **Open**. Expand the selected JAR file > **Resource Environment Providers** > *resource environment instance*. Right-click **Resource Environment Entries**, and click **New**. The following fields appear on the **General** tab:

Name:

Specifies the administrative name for the resource environment entry.

Description:

Specifies a description of the URL for your administrative records.

JNDI Name:

Specifies the Java Naming and Directory Interface (JNDI) name for the resource, including any naming subcontexts.

Use this name to link to the binding information of the platform. The binding associates the resources defined in the deployment descriptor of the module to the actual (or physical) resources bound into JNDI by the platform.

Custom Properties:

Specifies name-value pairs for setting additional properties on the object that is created at run time for this resource.

You must enter a name that is a public property on the object and a value that can be converted from a string to the type required by the set method of the property. The acceptable properties and values depend on the object that is created. Refer to the object documentation for a list of valid properties and values.

Example: Configuring Resource Environment settings:

You can configure Resource Environment settings. This topic provides the required fields and an example.

The purpose of this topic is to help you configure Resource Environment settings.

- Required fields:
  - Resource Environment Provider page: **Name**
  - Resource Environment Entry page: **Name, JNDI Name**
- Example:

```
<resources.env:ResourceEnvironmentProvider xmi:id="ResourceEnvironmentProvider_1"
name="resourceEnvProvider:name" description="resourceEnvProvider:description">
<classpath>resourceEnvProvider:classpath</classpath>
<factories xmi:type="resources.env:ResourceEnvEntry" xmi:id="ResourceEnvEntry_1"
name="resourceEnvEntry:name" jndiName="resourceEnvEntry:jndiName"
description="resourceEnvEntry:description">
<propertySet xmi:id="J2EEResourcePropertySet_20">
<resourceProperties xmi:id="J2EEResourceProperty_22"
name="resourceEnvEntry:customName" value="resourceEnvEntry:customValue"/>
</propertySet>
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_21">
<resourceProperties xmi:id="J2EEResourceProperty_23"
name="resourceEnvProvider:customName" value="resourceEnvProvider:customValue"/>
</propertySet>
</resources.env:ResourceEnvironmentProvider>
```

Example: Configuring resource environment custom settings for application clients:

You can configure resource environment custom settings.

The purpose of this topic is to help you configure resource environment custom settings.

- The custom page applies to every resource type. You can specify as many custom names and values as you need.
- Example:

```
<propertySet xmi:id="J2EEResourcePropertySet_20">
<resourceProperties xmi:id="J2EEResourceProperty_22"
name="resourceEnvEntry:customName" value="resourceEnvEntry:customValue"/>
</propertySet>
```

Running a Java EE client application with launchClient

After deploying a Java EE client application onto a machine with an Application Client installation or in a WebSphere Application Server node, you can start the application by using the launchClient command on that machine.

Before you begin

Before you can use the launchclient command to run a Java EE client application, you must have deployed the application.

This task only applies to Java EE client applications.

About this task

The Java Platform, Enterprise Edition (Java EE) specification requires support for a client container that runs Java applications (known as Java EE client applications) and provides Java EE services to the applications. Java EE services include naming, security, and resource connections.

Procedure

1. Start the Qshell environment.

   On the CL command line, type the command:

   ```
   STRQSH
   ```

2. Enter the following command to launch Java EE application clients:

   ```
   app_client_root/bin/launchClient
   ```

   where *app_client_root* is `/QIBM/ProdData/WebSphere/AppServer/V8/Base` or `/QIBM/ProdData/WebSphere/AppServer/V8/ND`.

3. Pass parameters to the `launchClient` command or to your application client program as well. The `launchClient` command allows you to do both. The `launchClient` command requires that the first parameter is either:
   - An EAR file specifying the application client to launch.
   - A request for `launchClient` usage information.

   The following example illustrates the command line invocation syntax for the launchClient tool:

   ```
   launchClient [-profileName pName | -JVMOptions options | -help | -?] userapp [-CCname=value] [app args]
   ```

   where

   - *userapp* is the path and the name of the EAR file that contains the application client.
   - *-CCname=value* is the client container name-value pair parameter. See the client container parameters section, for supported name-value pair arguments.
   - *app args* are arguments that pass to the application client.
   - *-profileName* defines the profile of the Application Server process in a multi-profile installation. The -profileName option is not required for running in a single profile environment or in an Application Clients installation.
   - *-JVMOptions* is a valid Java standard or non-standard option string. Insert quotation marks around the string.
   - *-help, -?* prints the usage information.

   All other parameters intended for the `launchClient` command must begin with the -CC prefix.

   Parameters that are not EAR files, or usage requests, or that do not begin with the -CC prefix, are ignored by the application client run time, and are passed directly to the application client program.

   The `launchClient` command retrieves parameters from three places:
   - The command line
   - A properties file
   - System properties

The parameters are resolved in the order listed above, with command line values having the highest priority and system properties the lowest. Using this prioritization you can set and override default values.

4. Specify the server name.

   By default, the launchClient command uses *your_server_name* for the `BootstrapHost` property value.

   This setting is effective for testing your application client when it is installed on the same computer as the server. However, in other cases override this value with the name of your server. You can override the `BootstrapHost` value by invoking `launchClient` command with the following parameters:

   ```
   launchClient myapp.ear -CCBootstrapHost=abc.midwest.mycompany.com
   ```

   You can also override the default by specifying the value in a properties file and passing the file name to the launchClient shell.

   Security is controlled by the server. You do not need to configure security on the client because the client assumes that security is enabled. If server security is not enabled, then the server ignores the security request, and the application client functions as expected.

Example

You can store launchClient values in a properties file, which is a good method for distributing default values. You can then override one or more values on the command line. The format of the file is one `launchClient -CC` parameter per line without the `-CC` prefix. For example:

```
 verbose=true classpath=/usr/lpp/mydir/util.jar;/usr/lpp/mydir/harness.jar;/usr/lpp
/production/G19/global.jar BootstrapHost=abc.westcoast.mycompany.com tracefile=/usr
/lpp/WebSphere/mylog.txt
```

launchClient tool:

This topic describes the Java Platform, Enterprise Edition (Java EE) command line syntax for the launchClient tool for WebSphere Application Server.

Important: All users who run commands from a specific profile must have authority to modify files that are created by other users that use the same profile. Otherwise, you might see a permission denied error in the log files. To avoid this issue, consider one of the following policies:

- Use specific profiles for distinct user authorities
- Always use the same user for all of the commands that are run in a given profile
- Ensure that all users of a specific profile belong to the same group. In addition, ensure that each user of a group has the read and write authority to the files that are created by other members in the same profile.

The following example illustrates the command line invocation syntax for the launchClient tool:

`launchClient [-profileName pName | -JVMOptions options | -help | -?] userapp [-CCname=value] [app args]`

where

- *userapp* is the path and the name of the EAR file that contains the application client.
- *-CCname=value* is the client container name-value pair parameter. See the client container parameters section, for supported name-value pair arguments.
- *app args* are arguments that pass to the application client.
- *-profileName* defines the profile of the Application Server process in a multi-profile installation. The -profileName option is not required for running in a single profile environment or in an Application Clients installation.
- *-JVMOptions* is a valid Java standard or nonstandard option string, except `-cp` or `-classpath`. Insert quotation marks around the string.
- *-help, -?* prints the usage information.

The first parameter must be -help, -? or contain no parameter at all. The -profileName pName and -JVMOptions options are optional parameters. If used, they must appear before the *<userapp>* parameter. All other parameters are optional and can appear in any order after the *userapp* parameter. The Java EE Application client run time ignores any optional parameters that do not begin with a -CC prefix and passes those parameters to the application client.

Client container parameters

Supported arguments include:

-CCadminConnectorHost
Specifies the host name of the server from which configuration information is retrieved.

The default is the value of the -CCBootstrapHost parameter or the value, your.server.name, if the -CCBootstrapHost parameter is not specified.

-CCadminConnectorPort
Indicates the port number for the administrative client function to use. The default value is 8880 for SOAP connections and 2809 for Remote Method Invocation (RMI) connections.

-CCadminConnectorType
Specifies how the administrative client connects to the server. Specify RMI to use the RMI connection type, or specify SOAP to use the SOAP connection type. The default value is SOAP.

-CCadminConnectorUser
Administrative clients use this user name when a server requires authentication. If the connection type is SOAP, and security is enabled on the server, this parameter is required.

-CCadminConnectorPassword
The password for the user name that the -CCadminConnectorUser parameter specifies.

-CCaltDD
The name of an alternate deployment descriptor file. This parameter is used with the -CCjar parameter to specify the deployment descriptor to use. Use this argument when a client JAR file is configured with more than one deployment descriptor. Set the value to null to use the client JAR file standard deployment descriptor.

-CCBootstrapHost
The name of the host server you want to connect to initially. The format is:
your_server_of_choice.com

-CCBootstrapPort
The server port number. If you do not specify this argument, the WebSphere Application Server default value is used.

-CCclassLoaderMode
Specifies the class loader mode. If PARENT_LAST is specified, the class loader loads classes from the local class path before delegating the class loading to its parent. The classes loaded for the following are affected:

- Classes defined for the Java EE application client
- Resources defined in the Java EE application
- Classes specified on the manifest of the Java EE client JAR file
- Classes specified using the -CCclasspath option

If PARENT_LAST is not specified, then the default mode, PARENT_FIRST, causes the class loader to delegate the loading of classes to its parent class loader before attempting to load the class from its local class path.

-CCclasspath
A class path value. When you launch an application, the system class path is used. If you want to

access classes that are not in the EAR file or part of the system class paths, specify the appropriate class path here. Multiple paths can be concatenated.

-CCD
Use this option to have the WebSphere Application Server set the specified system property during initialization. Do not use the equals (=) character after the -CCD. For example: -CCDcom.ibm.test.property=testvalue. You can specify multiple -CCD parameters. The general format of this parameter is -CCD<property key>=<property value>. For example, -CCDI18NService.enable=true.

-CCdumpJavaNameSpace
Controls generation of a dump of the java: name space for the application that is launched, which can be used for debugging purposes. A value of **true** generates a dump in short format, and includes the name and object type for each binding. A value of **long** generates a dump in long format, and includes additional information for each binding over short format, such as the local object type and string representation of the local object. The default value is **false**, and does not generate a dump.

-CCexitVM
Use this option to have the WebSphere Application Server call the System.exit() method after the client application completes. The default is false.

-CCinitonly
Use this option to initialize application client run time for ActiveX application clients without launching the client application. The default is false.

-CCjar
The name of the client Java Archive (JAR) file that resides within the EAR file for the application you wish to launch. Use this argument when you have multiple client JAR files in the EAR file.

-CCpropfile
Indicates the name of a properties file that contains launchClient properties. Specify the properties without the -CC prefix in the file, with the exception of the securityManager, securityMgrClass and securityMgrPolicy properties. See the following example: verbose=true.

-CCproviderURL
Provides bootstrap server information that the initial context factory can use to obtain an initial context. WebSphere Application Server initial context factory can use either a Common Object Request Broker Architecture (CORBA) object URL or an Internet Inter-ORB Protocol (IIOP) URL. CORBA object URLs are more flexible than IIOP URLs and are the recommended URL format to use. This value can contain more than one bootstrap server address. This feature can be used when attempting to obtain an initial context from a server cluster. You can specify bootstrap server addresses, for all servers in the cluster, in the URL. The operation will succeed if at least one of the servers is running, eliminating a single point of failure. The address list does not process in a particular order. For naming operations, this value overrides the -CCBootstrapHost and -CCBootstrapPort parameters. A CORBA object URL specifying multiple systems is illustrated in the following example:

```
-CCproviderURL=corbaloc:iiop:myserver.mycompany.com:9810,:mybackupserver.mycompany.com:2809
```

This value is mapped to the java.naming.provider.url system property.

-CCsecurityManager
Enables and runs the WebSphere Application Server with a security manager. The default is disable.

-CCsecurityMgrClass
Indicates the fully qualified name of a class that implements a security manager. Only use this argument if the -CCsecurityManager parameter is set to enable. The default is java.lang.SecurityManager.

-CCsecurityMgrPolicy
Indicates the name of a security manager policy file. Only use this argument if the -CCsecurityManager parameter is set to enable. When you enable this parameter, the java.security.policy system property is set. The default is *app_server_root*/properties/client.policy.

-CCsoapConnectorPort
The Simple Object Access Protocol (SOAP) connector port. If you do not specify this argument, the WebSphere Application Server default value is used.

-CCtrace
Use this option to obtain debug trace information. You might need this information when reporting a problem to IBM customer support. The default is `false`. For more information, read the Enabling trace topic.

-CCtracefile
Indicates the name of the file to which trace information is written. The default is to write output to the console.

-CCtraceMode
Specifies the trace format to use for tracing. If the valid value, `basic`, is not specified the default is advanced. Basic tracing format is a more compact form of tracing.

For more information on basic and advanced trace formatting, refer to the Interpreting trace output topic.

-CCverbose
This option displays additional information messages. The default is `false`.

If you are using an EJB client application with security enabled, edit the `sas.client.props` file, which is located in the *profile_root*`/properties` directory. Within the file, change the `com.ibm.CORBA.loginSource` value to *none*.

For more information on the `sas.client.props` utility, refer to the Manually encoding passwords in properties files and the PropFilePasswordEncoder command reference topics.

RMI connection with security. Used with the EJB and administrative client application.

Using Jacl:

```
wsadmin -conntype RMI -port  rmiportnumber -user userid
-password password
```

Using Jython:

```
wsadmin -lang jython -conntype RMI -port  rmiportnumber -user userid
-password password
```

rmiportnumber for your connection displays in the administrative console as BOOTSTRAP_ADDRESS.

Attention: On the AIX®, HP-UX, Linux, IBM i, Solaris, and z/OS operating systems, the use of -password option may result in security exposure as the password information becomes visible to the system status program, such as ps command, which can be invoked by other users to display all of the running processes. Do not use this option if security exposure is a concern. Instead, specify user and password information in the `soap.client.props` file for SOAP connector or `sas.client.props` file for RMI connector. The `soap.client.props` and `sas.client.props` files are located in the properties directory of your WebSphere Application Server profile.

If Kerberos (KRB5) is enabled for administrative authentication, the authentication target supports BasicAuth and KRB5. To use KRB5, update the `sas.client.props`, `soap.client.props`, and `ipc.client.props` files, according to the connector type.

Attention: When using Kerberos authentication, the user password does not flow across the wire. A one-way hash of password is used to identify the client.

The following examples demonstrate correct syntax.

```
/QIBM/ProdData/WebSphere/AppServer/V61/Base/bin/launchClient /home/earfiles/myapp.ear
   -profileName myprofile -CCBootstrapHost=myWASServer -CCverbose=true app_parm1 app_parm2
```

Specifying the directory for an expanded EAR file:

You can archive the `Manifest.mf` client Java Archive (JAR) files instead of automatically cleaning them up after the application exits.

Before you begin

Each time the launchClient tool is called, it extracts the Enterprise Archive (EAR) file to a random directory name in the temporary directory on your hard drive. Then the tool sets up the thread ClassLoader to use the extracted EAR file directory and JAR files included in the `Manifest.mf` client Java Archive (JAR) file. In a normal J2EE Java client, these files are automatically cleaned up after the application exits. This cleanup occurs when the client container shutdown hook is called. To avoid extracting the EAR file (and removing the temporary directory) each time the launchClient tool is called, complete the following steps:

Procedure

1. Specify a directory to extract the EAR file by setting the `com.ibm.websphere.client.applicationclient.archivedir` Java system property. If the directory does not exist or is empty, the EAR file is extracted normally. If the EAR file was previously extracted, the launchClient tool reuses the directory.
2. Delete the directory before running the launchClient tool again, if you need to update your EAR file. When you call the `launchClient` command, it extracts the new EAR file to the directory. If you do not delete the directory or change the system property value to point to a different directory, the launchClient tool reuses the currently extracted EAR file and does not use your changed EAR file. When specifying the `com.ibm.websphere.client.applicationclient.archivedir` property, make sure that the directory you specify is unique for each EAR file you use. For example, do not point the `MyEar1.ear` and the `MyEar2.ear` files to the same directory.

Downloading and running a Java EE client application using Java Web Start

Learn about the Java Web Start technology that is provided by the Java Standard Edition runtime environment to deploy Java Enterprise Edition application clients, including Thin application clients, on the remote client machine with a single click from a web browser on the client machine.

Before you begin

The supported client platforms for deploying application clients using the Java Web Start are the same as the IBM Application Client for WebSphere Application Server supported platforms, except Linux on Power® and OS/400® operating systems.

Before you begin this task, see the following topics to understand Java Web Start technology and its components:

- “Java Web Start architecture for deploying application clients” on page 143
- “Client application Java Network Launcher Protocol deployment descriptor file” on page 144
- “ClientLauncher class” on page 147

Note: The Sun Java Web Start, which is available from Sun Microsystems, is not compatible with the IBM Runtime Environment, Java 2 Technology Edition, which is provided by WebSphere Application Server and the IBM Application Client. The IBM Runtime Environment contains some additional functionality that is not supported in the Sun Java Web Start. Also, the IBM Runtime Environment uses a different packaging structure than the Sun Java Web Start. Use the IBM Runtime Environment.

About this task

To deploy application clients using Java Web Start, the client machine must have at least a Java SE runtime environment installed. The Java SE runtime environment includes the Java Web Start, which

implements the JSR 56: Java Network Launching Protocol and API. The application clients Enterprise Archive (EAR) file is a Java archive (JAR) resource in a JNLP descriptor file that resides on a central server. The JNLP descriptor file also specifies the runtime environment requirement for running the application.

WebSphere Application Server provides a launcher class to launch the Java EE application client in the application client container inside of Java Web Start. The client machine might not have the IBM Application Client for WebSphere Application Server installed. If this is the case, create and install an application client container and runtime package as a runtime environment through Java Web Start. The JNLP descriptor file specifies this runtime environment as the required runtime environment for running the Java EE application client.

WebSphere Application Server also provides command-line utility programs to create this application client container and runtime package from an existing IBM Application Client for WebSphere Application Server installation, as well as an installer class to install this package as a runtime environment for the application client container and also the Java Runtime Environment (JRE) in the IBM Application Client for WebSphere Application Server installation. To run the Java EE application client, the EAR file is deployed as a JAR resource that is described in the JNLP descriptor file.

Procedure

1. Identify the client machine operating system, and install the corresponding IBM Application Client for WebSphere Application Server on a development machine. For example, if the Java EE application clients are targeted to run on Windows operating systems, install the IBM Application Client for WebSphere Application Server for Windows.
2. Run the utility programs to create the application client container and runtime package.
   a. Use the “buildClientRuntime tool” on page 152 utility to create the package.
   b. Use the “buildClientLibJars tool” on page 144 utility to create the JAR files containing the launcher and the installer class. This utility also zips up the properties files in the <app_client_root>/properties directory.
3. Create the runtime installer JNLP descriptor file. The JNLP response must be included in the JNLP version ID to indicate the current runtime version in the response header, for example, `x-java-jnlp-version-id=1.6.0`. Using a servlet of a JavaServer Pages (JSP) file to provide a dynamic JNLP response.
4. Create the Java EE application client launch JNLP descriptor file.
5. Package the application client container runtime environments and the Java EE application in an Enterprise Archive (EAR) file. Depending on your preferred deployment strategy, the files can be in two separate Web modules, or combined into one.
6. All JAR resources must be Java signed, including the Java EE application client EAR file.
7. Deploy the Enterprise Archive file on an application server, and start the application. The Java EE application client is ready to be deployed.

Example

A Java Web Start deployment Sample is included in the client samples. This Sample demonstrates the steps to deploy a Java EE application client with an automated ANT script. The Sample has a servlet to generate the runtime installer JNLP response with JNLP version ID, for example, `x-java-jnlp-version-id`.

Important: When the application client initially launches using Java Web Start from Sun Microsystems Java SE Runtime Environment 6.0, it installs the Application Client runtime, which includes the IBM JRE. An null pointer exception (NPE) is thrown from the `com.sun.deploy.services.WPlatformService.getSecureRandom()` method. This is a known bug in Sun Java SE 6 (http://bugs.sun.com/bugdatabase/view_bug.do?bug_id=6505528). If you experience this exception, relaunch the application. The NPE only occurs on the first launch of the application client.

Java Web Start architecture for deploying application clients:

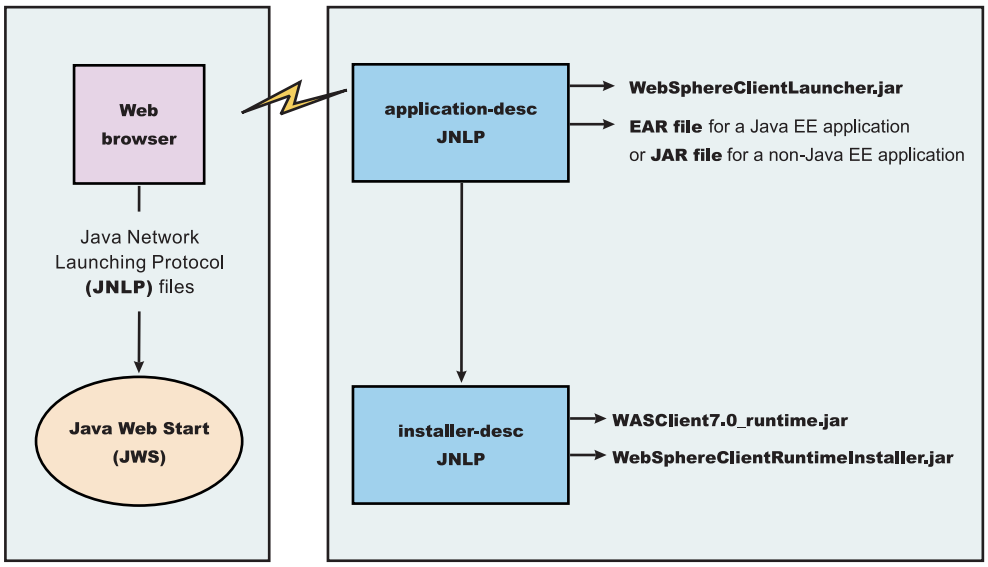
Java Web Start is an application-deployment technology that includes the portability of applets, the maintainability of servlets and JavaServer Pages (JSP) file technology, and the simplicity of mark-up languages such as XML and HTML. It is a Java application that allows full-featured Java EE client applications to be launched, deployed and updated from a standard Web server. The Java Web Start client is used with platforms that support a web browser.

Java Web Start is not supported.

Upon launching Java Web Start for the first time, you might download new client applications from the Web. Each time you launch JWS thereafter, you can initiate applications either through a link on a web page or (in Windows) from desktop icons or the Start menu. You can deploy applications quickly using Java Web Start, cache applications on the client machine, and launch applications remotely offline. Additionally, because Java Web Start is built from the Java Platform, Enterprise Edition (Java EE) infrastructure, the technology inherits the complete security architecture of the Java EE platform.

The technology underlying Java Web Start is the Java Network Launching Protocol & API (JNLP). Java Web Start is a JNLP client and it reads and parses a JNLP descriptor file (JNLP file). Based on the JNLP descriptor, it downloads appropriate pieces of a client application and any of its dependencies. If any of the pieces of the application are already cached on the client machine, then those components are not downloaded again, unless they have been updated on the server machine. After you download and cache the client application, JWS launches it natively on the client machine.

The following diagram shows an overview of launching a client application, include the Application Client for WebSphere Application Server as a dependent resource, using Java Web Start.



The web browser running on a client machine connects to a web application located on a server machine. The client application JNLP descriptor file is downloaded and processed by Java Web Start on the client machine.

In this diagram, there are two JNLP descriptor files:

- Client application JNLP descriptor (application-desc in the diagram)
- Application Clients run-time installer JNLP descriptor (installer-desc in the diagram)

Each of these JNLP descriptor files, the client application (JAR or EAR) and the dependent resource JAR files are packaged as web applications in an EAR file. This EAR file is deployed to an Application server.

The client machine with JWS installed uses a web browser to connect to the URL of the client application JNLP descriptor file to download and run the client application.

Using Java Web Start from Java SE Runtime Environment 6.0 or later is highly recommended. All the platforms supported by the application client for WebSphere Application Server are supported with the exception Linux on Power and OS/400 platforms.

You can use the following:

- Java Web Start on the Java Standard Edition Developer Kits that IBM provides, packaged in Application Client for WebSphere Application Server
- Java Web Start on Sun Microsystems Java SE 6 Development Kit or Java SE Runtime Environment 6.0, which you can download from the Sun Microsystems website for Windows, Linux and Solaris operating systems
- Java Web Start on HP-UX JDK or JRE for Java Platform, Standard Edition, Version 6, which you can download from the HP website

buildClientLibJars tool:

For a Java Platform, Enterprise Edition (Java EE) application client application and or Thin application client application to be launched using Java Web Start (JWS), the properties files bundled in Application Client for WebSphere Application Server must be installed in the Java Web Start. Use this tool to create those property JAR files. The Java Web Start client is used with platforms that support a web browser.

Java Web Start is not supported on WebSphere Application Server for IBM i.

The buildClientLibJars tool copies the JAR files from the Application Client for WebSphere Application Server installation and creates a `properties.jar` file, which contains the properties files from the Application Clients installation properties directory to a specified location. When this property is created, the tool uses the value of `keystore`, `storepass`, `alias` and `storetype` to sign all of the JAR files in the specified location.

```
Windows usage: buildClientLibJars.bat  [-help] [-verbose] destdir keystore storepass alias storetype

Unix usage:    buildClientLibJars.sh  [-help] [-verbose] destdir keystore storepass alias storetype
```

where:

- `-help` will display the message
- `-verbose` will turn on verbose message
- `destdir` will output the destination directory name
- `keystore` is the key store file
- `storepass` is the key store password
- `alias` key is the alias name
- `storetype` is the key store type

Client application Java Network Launcher Protocol deployment descriptor file:

The deployment descriptor file is the main Java Network Launcher Protocol (JNLP) descriptor file for the client application.

Location

The client application has an Application Clients runtime dependency that provides the following:

- Java SE Runtime Environment from IBM
- Application Clients run-time properties

- SSL KeyStore and TrustStore file
- Application Clients run-time library JAR files (optional for Thin Application client applications)

If the Application Clients run-time dependency is not met, it is downloaded and installed in Java Web Start (JWS), as described by the Application Clients run-time installer JNLP descriptor file. For example:

```
<j2se version="1.6" href="http://your_server.com/jws/wasappclient/download.jnlp"/>
```

Usage notes

The client application must also include the `WebSphereClientLauncher.jar` file, which contains the launcher class, com.ibm.websphere.client.launcher.ClientLauncher, that completes one of the following actions:

- If it is a Java Platform, Enterprise Edition (Java EE) Application client application (that is the resources for the application contain an EAR file with a client application), the EAR file must be specified as a JAR resource so that it can be downloaded to JWS and specified in the system property, `com.ibm.websphere.client.launcher.ear`. See “JNLP descriptor file for a Java EE Application client application” on page 146 for an example.
- If it is a Thin Application client application, the Thin Application client application JAR file must be specified as a JAR resource so that it can be downloaded to JWS and the name of the class containing main method entry point is specified in the system property, `com.ibm.websphere.launcher.main`. See “JNLP descriptor file for a Thin Application client application” on page 146 for an example.

The JNLP specification requires all the resource (JAR or EAR) files used in a JNLP file to be signed.

You can specify the –CC arguments defined in the launchClient tool for a J2EE Application client application in application arguments section of the JNLP descriptor files. However, only –CCD is supported for a Thin Application client application to define system properties and the JNLP <property> tag can also be used to define system properties. See the following example for details:

```
<property name="java.naming.provider.url" value="corbaloc:iiop:myserver.com:9089"/>
```

For a J2EE Application client application, specify the following application arguments as defined in the JNLP.

1. Specify your target server provider URL, as shown in the following example:

   ```
   <argument> >-CCDjava.naming.provider.url =corbaloc:iiop:myserver.mydomain.com:9080 </argument>
   ```

2. Specify the SSL Key File and SSL Trust File location. These files are expected to be available in the client machine. To use the ones in the Application Clients run-time dependency installed in JWS cache, specify these application arguments:

   ```
   <argument> -CCDcom.ibm.ssl.keyStore=${WAS_ROOT}/etc/key.p12 </argument>
   <argument> -CCDcom.ibm.ssl.trustStore=${WAS_ROOT}/etc/trust.p12 </argument>
   ```

3. Specify the initial naming context factor, as shown in the following example:

   ```
   <argument>-CCDjava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory </argument>
   ```

   For a Thin Application client application, you also need to specify the actual location of the `sas.client.props` and `ssl.client.props` files located in the Application Clients runtime dependency that is installed in the JWS cache.

   ```
   <argument>-CCDcom.ibm.CORBA.ConfigURL=file:${WAS_ROOT}/properties/sas.client.props </argument>
   <argument>-CCDcom.ibm.SSL.ConfigURL=file:${WAS_ROOT}/properties/ssl.client.props </argument>
   ```

   If any of the default settings in the `sas.client.props` and `sas.client.props` file need modifying, use the –CCD to change the settings through the system properties, as shown in the following example:

   ```
   <argument>-CCDjavacom.ibm.CORBA.securityEnabled=false </argument>
   ```

   Important: The ${WAS_ROOT} token used in the JNLP file is replaced by the launcher class, com.ibm.websphere.client.launcher.ClientLauncher, to the actual location of the Application Clients run-time dependency installation in the JWS cache. If you are using JSP to dynamically create this JNLP description file, you must escape this token because it has a different meaning in JSP 2.0. See the following example for details:

```
<argument>-CCDcom.ibm.ssl.keyStore=\${WAS_ROOT}/etc/key.p12 </argument>
<argument>-CCDcom.ibm.ssl.trustStore=\${WAS_ROOT}/etc/trust.p12 </argument>
```

JNLP descriptor file for a Java EE Application client application:

The deployment descriptor file is the main Java Network Launcher Protocol (JNLP) descriptor file for the client application.

Here is an example of the client application JNLP descriptor file for a Java EE Application client application:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
This sample program is provided AS IS and may be used, executed, copied and modified
without royalty payment by customer (a) for its own instruction and study, (b) in order
to develop applications designed to run with an IBM WebSphere product, either for customer's
own internal use or for redistribution by customer, as part of such an application, in
customer's own products.

Licensed Materials - Property of IBM

5724-I63, 5724-H88, 5724-H89, 5655-N02, 5724-J08

Copyright IBM Corp. 2008 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
-->

<jnlp spec="1.0+" codebase="http://your_server:port_number/jws/wasappclient/apps/">
 <information>
  <title>Java EE Client Example</title>
  <vendor>IBM</vendor>
  <homepage href="null"/>
  <description>Java WebStart example: Launching Java EE Application Client</description>
  <description kind="short">Java EE Applicaiton Client</description>
  <description kind="tooltip">Java EE Application Client</description>
 </information>

 <security>
  <all-permissions/>
 </security>

 <resources>
  <j2se href="http://your_server:port_number/jws/wasappclient/JREDownload.xjnlp" version="1.6"/>
  <jar href="../lib/WebSphereClientLauncher.jar" download="eager" main="false"/>
  <jar href="../lib/properties.jar" download="eager" main="false"/>
  <jar href="SwingCalculator.ear" download="eager" main="false"/>

  <property name="com.ibm.websphere.client.launcher.ear" value="SwingCalculator.ear"/>
 </resources>

 <application-desc main-class="com.ibm.websphere.client.launcher.ClientLauncher">
  <argument>-CCproviderURL=corbaloc:iiop:tiu03.torolab.ibm.com:2809</argument>
 </application-desc>

</jnlp>
```

JNLP descriptor file for a Thin Application client application:

The deployment descriptor file is the main Java Network Launcher Protocol (JNLP) descriptor file for the client application. If it is a Thin Application client application, then the launcher class uses the current JVM from the Application Clients run-time dependency and invokes the Thin Application client application main method.

Here is an example of the JNLP descriptor file for a Thin Application client application.

```
 This sample program is provided AS IS and may be used, executed, copied and modified
without royalty payment by customer (a) for its own instruction and study, (b) in order
to develop applications designed to run with an IBM WebSphere product, either for customer's
own internal use or for redistribution by customer, as part of such an application, in
customer's own products.

Licensed Materials - Property of IBM

5724-I63, 5724-H88, 5724-H89, 5655-N02, 5724-J08

Copyright IBM Corp. 2008 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.

Licensed Materials - Property of IBM

5724-I63, 5724-H88, 5724-H89, 5655-N02, 5724-J08

Copyright IBM Corp. 2008 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with
IBM Corp.
-->

<!-
========================================================================
->
<!-- TODO: change "codebase" to the actual URL location of the jnlp file ->
========================================================================
->
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.0+"
codebase="http://your_server:port_number/jws/wasappclient/apps">
  <information>
  <title>Thin Base Calculator Client Samples</title>
  <vendor>IBM</vendor>
  <description>Thin Base Calculator Client Samples</description>
  <offline-allowed/>
 </information>

 <security>
  <all-permissions/>
 </security>
  <resources>
   <j2se version="1.6" href="http://your_server:port_number/jws/wasappclient/JREDownload.xjnlp"/>

   <jar href="/jws/wasappclient/lib/WebSphereClientLauncher.jar" main="true"/>
   <jar href="BasicCalculatorClientCommon.jar"/>
   <jar href="BasicCalculatorEJB.jar"/>
   <jar href="BasicCalculatorThinClient.jar"/>

   <property name="com.ibm.websphere.client.launcher.main"
      value="com.ibm.websphere.samples.technologysamples.basiccalcthinclient.BasicCalculatorClientThinMain"/>
   <property name="java.naming.factory.initial"
              value="com.ibm.websphere.naming.WsnInitialContextFactory" />
   <property name="java.naming.provider.url"
               value="corbaloc:iiop:tiu03:2809"/>

   </resources>

   <add</argument>
    <argument>1</argument>
    <argument>2</argument>
   </application-desc>
</jnlp>
```

ClientLauncher class:

The class, com.ibm.websphere.client.installer.ClientLauncher, contains a main() method that is called by Java Web Start (JWS) to launch the client application. The Java Web Start client is used with platforms that support a web browser.

Java Web Start is not supported.

This client is packaged in the `WebSphereClientLauncher.jar` file that is located in the Application Client for WebSphere Application Server installation under the <*app_client_root*>/ lib/webstart directory.

The launcher class requires that the following properties are defined. These properties are not defined in a separate properties file. Instead, the properties are defined as part of the Java Network Launching Protocol (JNLP) files.

com.ibm.websphere.client.launcher.main
If the client application is a Thin Application client, then this property should be specified. It specifies the class where the main entry point of the client application resides.

com.ibm.websphere.client.launcher.ear
If the client application is a Java Platform, Enterprise Edition (Java EE) Application client, then this property should be specified. It specifies the name of the EAR file to be executed. This property takes precedence over com.ibm.websphere.client.launcher.main. However, only one of the two properties should be specified.

Application client launcher for Java Web Start:

The application client launcher for Java Web Start is a Java class, com.ibm.websphere.client.installer.ClientLauncher, which has a main() method that Java Web Start calls to start the application client container and to invoke the application client's main() method. It provides similar functions as the **lauchClient** command line tool to start application clients from the command line.

The com.ibm.websphere.client.launcher.ClientLauncher class is packaged in the WebSphereClientLauncher.jar file under the <app_client_root>/lib/webstart directory.

The launcher tool requires that the following properties are defined.

com.ibm.websphere.client.launcher.main
If the client that is to be run is a thin client, then this property should be specified. It specifies the class where the main entry point of the application resides. It is the main class name for a Thin application client. If it is set, the launcher will not start the client container, it will rather invoke the main method for the application directly. However, if `com.ibm.websphere.client.launcher.ear` is also set, it will be ignored.

com.ibm.websphere.client.launcher.ear
If the client that is to run is the Java Platform, Enterprise Edition (Java EE) client, then this property should be specified. It specifies the name of the ear file to be executed. This property takes precedence over `com.ibm.websphere.client.launcher.main` although only one of the two properties should be specified.

These properties are not defined in a separate properties file. Instead, they are defined as part of the Java Network Launching Protocol files.

When `com.ibm.websphere.client.launcher.ear` is set, the application client launcher for JWS supports almost all of the –CC arguments as the **lauchClient** command line tool supports. However, if only `com.ibm.websphere.client.launcher.main` is set, the launcher will only support the –CCD argument. The following table shows the comparison of the supported –CC arguments for the **launchClient** command line tool and the application client launcher for JWS:

Table 14. Comparison of the supported –CC arguments for the **launchClient** *command line tool and the application client launcher for JWS. Comparison of the supported –CC arguments*

-CC argument	launchClient	Application client launcher for JWS
-CCverbose	Yes	Yes
-CCjar	Yes	Yes
-CCclasspath	Yes	N/A
-CCadminConnectorHost	Yes	Yes
-CCadminConnectorPort	Yes	Yes
-CCadminConnectorType	Yes	Yes
-CCadminConnectorUser	Yes	Yes
-CCaltDD	Yes	Yes
-CCBootstrapHost	Yes	Yes
-CCBootstrapPort	Yes	Yes
-CCproviderURL	Yes	Yes
-CCinitonly	Yes	N/A
-CCtrace	Yes	Yes
-CCtracefile	Yes	Yes
-CCsecurityManager	Yes	N/A
-CCsecurityMgrClass	Yes	N/A
-CCsecurityMgrPolicy	Yes	N/A
-CCD	Yes	Yes
-CCexitVM	Yes	Yes
-CCdumpJavaNameSpace	Yes	Yes
-CCsoapConnectorPort	Yes	Yes
-CCtraceMode	Yes	Yes
-CCclassLoaderMode	Yes	Yes

Macro expansion is supported for the –CCD argument by the application client launcher for JWS. The launcher will automatically substitute certain macro keys (enclosed with ${...}) with the calculated value at runtime. For example, if a macro key is used in the –CCD argument in the application client JNLP manifest file,

```
<argument>-CCDcom.ibm.ssl.keyStore= ${WAS_ROOT}/etc/key.p12</argument>
```

it will be expanded to the JWS cache installation root location and the argument will become:

```
-CCDcom.ibm.ssl.keyStore=/home/tiu/.java/deployment/cache/javaws/ext/E1134532441112/etc/key12.p12
```

The following table shows the three macro keys that are currently supported and will be substituted by the launcher:

Table 15. Currently supported macro keys. Supported macro keys

Macro key	Value
${WAS_ROOT}	Installation root location within the JWS cache that is used by the application client container and runtime installer for JWS.
${JAVA_HOME}	Location of Java home. The return value of System.getProperty(“java.home”).

Table 15. Currently supported macro keys (continued). Supported macro keys

Macro key	Value
${USER_HOME}	Location of user home. The return value of System.getProperty("user.home").

Preparing the application client run time dependency component for Java Web Start:

To launch a Java Platform, Enterprise Edition (Java EE) application client application, a Thin application client application, or both using Java Web Start (JWS), a Java Runtime Environment implementation Java archive (JAR) that IBM provides, the library JAR files and properties files bundled in Application Client for WebSphere Application Server must be installed in the JWS. Learn the steps to build the application client run time dependency component from an application client installation. It is packaged as a web application archive (WAR) file that can be installed in an application Server.

Before you begin

Install the Application Client for WebSphere Application Server for the operating system to which the client application deploys. If there is a requirement to deploy the client application to multiple operating systems, the application client run time dependency component must be built separately for each operating system that client application supports.

Procedure

1. Install the Application Client for WebSphere Application Server for the client application supported operating systems.
2. Change the directory to the installation bin directory.
3. Run the "buildClientRuntime tool" on page 152 to generate the application client run time JAR file, which contains the Java Standard Edition Runtime Environment, the run time library JAR files, properties files, and the SSL KeyStore and TrustStore files from the application client installation.
4. Run the buildClientLibJars tools to package up the properties files in the properties directory of the application client installation into a properties.jar file in the specified location. The buildClientLibJars tools will also copy the `WebSphereClientLauncher.jar` file and `WebSphereClientRuntimeInstaller.jar` file from the application client installation to the specified location. All jar files in the specified location will be signed by the provided certificate.

   For example, if you are using Version 7.0 and using the test certificate that is included in the application client installation:

   ```
   buildClientLibJars C:\Temp\webstart ..\etc\DummyClientKeyFilejar WebAS "websphere dummy client" JKS
   ```
5. Create a JavaServer Pages (JSP) file or use a servlet to generate the application client run time installer Java Network Launching Protocol (JNLP) descriptor to respond to Java Web Start request. See the Java Web Start deployment sample in the application client installation.
6. Package the two signed JAR files, `WASClient7.0_windows.jar` and `WebSphereClientRuntimeInstaller.jar`, and the JSP file or servlet for generating the Application Client run time installer JNLP descriptor into a web application archive (WAR) file. This WAR file is packaged into an EAR file that can be deployed to an application server. See the Java Web Start deployment sample in the application client installation.

Results

Your web application is ready to serve the application client run time and the JRE environment.

Example

```
<!-- This sample program applies to WebSphere Application Server, Version 6.1.
It is provided AS IS and may be used, executed, copied and modified
without royalty payment by customer (a) for its own instruction and study, (b) in order
```

```
to develop applications designed to run with an IBM WebSphere product, either for customer's
own internal use or for redistribution by customer, as part of such an application, in
customer's own products.

Product 5630-A36,  (C) COPYRIGHT International Business Machines Corp., 2005
All Rights Reserved * Licensed Materials - Property of IBM
-->

<%-- // to set the Last_Modified header so that the JNLP client will know whether to download
     // the JNLP file again and update the cached copy.
     String jspPath = application.getRealPath(request.getServletPath());
     java.io.File jspFile = new java.io.File(jspPath);
     long lastModified = jspFile.lastModified();
%><%
      // locally declared variables
      String url=request.getRequestURL().toString();
      String jnlpCodeBase=url.substring(0,url.lastIndexOf('/'));
      String jnlpRefURL=url.substring(url.lastIndexOf('/')+1,url.length());

      // Need to set a JNLP mime type - if WebStart is installed on the client,
      // this header will induce the browser to drive the WebStart Client
      response.setContentType("application/x-java-jnlp-file");                          1
      response.setHeader("Cache-Control", null);
      response.setHeader("Set-Cookie", null);
      response.setHeader("Vary", null);
      response.setDateHeader("Last-Modified", lastModified);

      // An installer must reply with the version number for a given install
      if (response.containsHeader("x-java-jnlp-version-id"))
        response.setHeader("x-java-jnlp-version-id", "WASClient6.1.0");            2
      else
        response.addHeader("x-java-jnlp-version-id", "WASClient6.1.0");
%>

<?xml version="1.0" encoding="utf-8"?>

<!-- ================================================================ -->
<!-- TODO: change "codebase" to the actual url location of this jsp -->
<!-- ================================================================ -->

<jnlp spec="1.0+"
codebase="http://YOUR_APP_SERVER:PORTNUMBER/WEBAPP_CONTEXT_ROOT/Runtime/WebSphereJre">

<information>
  <title>Application Client Java Runtime Environment</title>
  <vendor>IBM</vendor>
  <icon href="icon.gif"/>
  <description>Application Client Java Runtime Environment</description>
  <description kind="short">Application Client JRE</description>
  <description kind="tooltip">Application Client JRE</description>
  <offline-allowed/>
</information>

<security>
 <all-permissions/>
</security>

 <resources>
  <j2se version="1.4+"/><%-- The installer can use any 1.4 JRE --%> 3
   <jar href="WebSphereClientRuntimeInstaller.jar" main="true"/> 4

  <!-- JRE version registration with Web Start -->
  <property name="com.ibm.websphere.client.jre.version" value="WASClient6.1.0"/> 5
```

```
 </resources>

 <resources os="Windows"> 6
<!-- ============================================================ -->
<!-- TODO: the property value for unix platform is "java/jre/bin/javaw" -->
<!--   and the "os" value match to your target client machine platform             -->
<!-- ============================================================ -->

   <jar href="WASClient6.1.0_Windows.jar"/> 7

<!-- ============================================================ -->
<!-- TODO:  property value for unix platform is "java/jre/bin/javaw" -->
<!-- ============================================================ -->
<!-- relative path of the jre executable —->

   <property name="com.ibm.websphere.client.jre.launch.java"
 value="java\jre\bin\javaw.exe"/> 8

 </resources>
 <installer-desc main-class="com.ibm.websphere.client.installer.ClientRuntimeInstaller"/>
</jnlp>
```

1. Specifies that the file is a JNLP mime type so that the browser can process the JNLP file.
2. Specifies the exact version of this Application Client run time dependency component in the response by setting the HTTP header field: `x-java-jnlp-version-id`.
3. Specifies the required JRE version to run the installer program.
4. Specifies the installer `WebSphereClientRuntimeInstaller.jar` file, which contains the ClientRuntimeInstaller class.
5. Specifies a system property that defines the version of Application Client run time dependency component. This version is registered to the JNLP client.
6. Specifies resources for a particular platform. Each supported client application platform needs its own separate JAR file.
7. Specifies the Application Client run time dependency component JAR file.
8. Specifies the program to call that starts a JVM for the client application.

buildClientRuntime tool:

For a Java Platform, Enterprise Edition (Java EE) application client application and or Thin application client application to be launched using Java Web Start (JWS), the library JAR files bundled in Application Client for WebSphere Application Server must be installed in the Java Web Start. Use this tool to build those JAR files. The Java Web Start client is used with platforms that support a web browser.

The Java Web Start client is not supported on WebSphere Application Server for OS/400.

The buildClientRuntime tool builds the required components from the WebSphere Application Server clients installation into the JAR file specified on the command. This JAR file contains:

- License files
- Java SE Runtime Environment 6 (JRE 6) that IBM provides
- Application Clients runtime properties and configuration
- SSL KeyStore and TrustStore files
- Runtime library JAR files

In the case of building an Application Clients runtime JAR file only for serving Thin Application client applications and not for Java EE Application client applications, the runtime library JAR files and the Application Clients runtime properties files are not included, except the configuration files,

sas.client.props, ssl.client.props and soap.client.props, located in the WAS_ROOT/properties directory. The Java Web Start client is used with platforms that support a web browser.

The command-line invocation syntax for the buildClientRuntime tool is shown in the following example:

```
Windows Usage:    buildClientRuntime.bat  [-help] [-verbose] outfile keystore storepass alias storetype

Unix Usage:         buildClientRuntime.sh    [-help] [-verbose] outfile keystore storepass alias storetype
```

where:

- -help will display the message
- -verbose will turn on verbose message
- outfile is the output file name
- keystore is the key store file
- storepass is the key store password
- alias is the key alias name
- storetype is the key store type

ClientRuntimeInstaller class:

This section provides information on the ClientRuntimeInstaller class.

This class, com.ibm.websphere.client.installer.ClientRuntimeInstaller, contains a main() method that Java Web Start (JWS) calls to install the Application Client for WebSphere Application Server run-time dependency component in JWS cache. It is packaged in WebSphereClientRuntimeInstaller.jar file located in the Application Client for WebSphere Application Server installation in the *<app_server_root>*/JWS directory.

Specify the WebSphereClientRuntimeInstaller.jar file and the Application Client run-time dependency component JAR file as JAR resources in the Application Client run-time installer Java Network Launcher Protocol (JNLP) descriptor file. See the following example for details:

```
<jar href="Launcher/WebSphereClientRuntimeInstall.jar" main="true"/>
<jar href="Launcher/WASClient6.1_windows.jarRuntimeInstall.jar" main="true"/>
```

The ClientRuntimeInstaller class main method requires the following properties to be set in the JNLP file:

com.ibm.websphere.client.jre.version
: Specifies a Java Runtime Environment (JRE) version name that is to be used when referring to the Application Client run-time dependency component.

com.ibm.websphere.client.jre.launch.java
: Specifies the relative location of the javaw.exe program in the Application Client run-time dependency component JAR file.

The previously mentioned properties, JRE version name and the location of the javaw.exe program are registered to the Java Web Start Application Manager, as shown in the following example:

```
<property name="com.ibm.websphere.client.jre.version" value="WASclient6.1"/>
<property name="com.ibm.websphere.client.jre.launch.java" value="java\jre\bin\javaw.exe"/>
```

Using the Java Web Start sample:

The EAR file, WasAppClientRuntime.ear, is provided in the *app_client_root*/samples/bin/WasAppClientRuntime directory of the Client Application for WebSphere Application Server installation. This EAR file provides a sample Application Clients run-time installer JNLP descriptor file and a sample Application Clients run-time library component JNLP descriptor file. Follow the steps in this task to build the Application Clients run-time dependency component and the Application Clients run-time library

component. Add these components to the WebSphereClientRuntime.ear file, and then install the EAR file in an Application Server to be used by the client application.

About this task

There is a new Java Web Start sample available in the client sample gallery for WebSphere Application Server V7.0. Refer to the client sample gallery in the Application Client for WebSphere Application Server product. The name of the new sample is "Java Web Start Deployment Sample".

Installing Java Web Start:

Learn about the steps that are necessary to install Java Web Start (JWS).. Java Web Start technology is provided by the Java SE runtime environment to deploy Java EE application clients (including Thin application clients) on the remote client machine with a single click from a web browser on the client machine.

Before you begin

Before you begin this task, see the “Preparing the application client run time dependency component for Java Web Start” on page 150 topic to understand Java Web Start (JWS) technology and components.

Note: The Sun Java Web Start, which is available from Sun Microsystems, is not compatible with the IBM Runtime Environment, Java 2 Technology Edition, which is provided by WebSphere Application Server and the IBM Application Client. The IBM Runtime Environment contains some additional functionality that is not supported in the Sun Java Web Start. Also, the IBM Runtime Environment uses a different packaging structure than the Sun Java Web Start. Use the IBM Runtime Environment.

About this task

Complete the following steps to install JWS:

Procedure

1. Install IBM Application Client for WebSphere Application Server.
2. Change your directory to the javaws path.
   - *client_install_root*\java\jre\lib\javaws
3. Run the update settings script from the path mentioned in the previous step.
   - Run the updateSetting.sh script
4. Change your path to the JWS installed path. For example, enter:
   - *client_install_root*\java\jre\javaws
5. Run javaws from the path mentioned in the previous step.
   - Run ./javaws command.

Using a static JNLP file with Java Web Start for Application clients:

Do not use JSP to dynamically generate a JNLP file, otherwise the JNLP jsp page cannot be opened in some IE browsers.

About this task

To use a static JNLP file, you will need to add the following mime type mapping in the web.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">
       <display-name>
       WAS Client runtime for Java Web Start</display-name>
       <welcome-file-list>
              <welcome-file>index.html</welcome-file>
              <welcome-file>index.htm</welcome-file>
              <welcome-file>index.jsp</welcome-file>
              <welcome-file>default.html</welcome-file>
              <welcome-file>default.htm</welcome-file>
              <welcome-file>default.jsp</welcome-file>
       </welcome-file-list>
       <mime-mapping>
       <extension>jnlp</extension>
       <mime-type>application/x-java-jnlp-file</mime-type>
       </mime-mapping>
</web-app>
```

Running the IBM Thin Client for Enterprise JavaBeans (EJB)

An EJB Client is a Remote Method Invocation over Internet Inter-ORB Protocol (RMI-IIOP) Java Platform, Standard Edition (Java SE) application that accesses remote Enterprise Java Beans from a server through Java Naming and Directory Interface (JNDI) look up. IBM Thin Client for EJB offers a smaller footprint and is easy to deploy to a Java SE environment and an Eclipse Rich Client Platform (RCP) environment. You can bundle the IBM Thin Client for EJB library using the WebSphere Application Server installation or the Application Client for WebSphere Application Server installation with your application. The IBM Thin Client for EJB also extends the choice of Java SE runtime. It can be run in the Java Runtime Environment (JRE) that is packaged with the WebSphere Application Server product, the Sun Microsystems JRE that is downloaded from the Sun Microsystems website, or the JRE that is downloaded from the HP website.

Before you begin

The IBM ORB implementation library is required if the IBM Thin Client for EJB is running with a non-IBM product JRE on a non-IBM product platform. For example, running the IBM Thin Client for EJB with Sun Microsystems JRE on Windows, Linux, or Solaris, and with the HP JRE on HP-UX. The IBM-provided Solaris hybrid and HP hybrid JRE are not considered non-IBM product JRE environments.

The IBM Thin Client for EJB can access version 2.x and version 3.x EJB on the WebSphere Application Server using the JNDI lookup, but it cannot access version 3.x EJB through resource injection. Resource injection is supported if the client application is a Java Platform, Enterprise Edition (Java EE) Application Client running within the Java Platform, Enterprise Edition (Java EE) Application Client Container.

Before you set up an EJB Thin Client environment, obtain the Java archive (JAR) file for the EJB Thin Client for WebSphere Application Server. To obtain the EJB Thin Client for WebSphere Application Server, install WebSphere Application Server or Application Client. The EJB Thin Client for WebSphere Application Server file, `com.ibm.ws.ejb.thinclient_8.0.0.jar`, is located in the *app_server_root*`\runtimes` directory.

Copy the Java archive (JAR) file for the IBM Thin Client for EJB with WebSphere Application Server product, `com.ibm.ws.ejb.thinclient_8.0.0.jar` and the `endorsed_apis_8.0.0.jar` files, to other machines to create a lightweight client environment that enables communications with the products. Copies of the IBM Thin Client for EJB are subject to the same terms and conditions of the license agreement for the WebSphere product where you obtained the Thin Client for EJB. Refer to the license agreements for correct usage and other limitations.

Copy the *app_server_root*`\runtimes\endorsed\endorsed_apis_8.0.0.jar` file into the default directory, `JAVA_JRE\lib\endorsed`. Alternatively, you can use the java.endorsed.dirs property to specify a directory of your choice. If you choose to use an alternative directory, it is a best practice to only include the endorsed_apis JAR file.

The IBM Thin Client for EJB with WebSphere Application Server runs on distributed operating systems with JDK support, including both Version 5 and Version 6. When using the IBM Thin Client for EJB as a standalone Java SE application with a non-IBM product JRE, you must override the default ORB implementation for the JRE through one of following methods:

- Include the com.ibm.ws.orb_8.0.0.jar file in the Java system classpath.
- Override the default ORB implementation in the JRE, using Java Endorsed Standards Override Mechanism.
- Set the java.endorsed.dirs path to a directory that contains the com.ibm.ws.orb_8.0.0.jar file.

When running the IBM Thin Client for EJB as an Eclipse RCP application, it is recommended to use method two, to override the default JRE ORB implementation.

Important: The Pluggable Application Client is deprecated. It is replaced by the IBM Thin Client for EJB.

Attention: When running the IBM Thin Client for EJB, and the `-Djava.util.logging.manager=com.ibm.ws.bootstrap.WsLogManager` command line option is used, a ClassDefNotFoundError error is thrown. The use of WsLogManager is not supported in the IBM Thin Client for EJB, but you can use another Java logging manager.

About this task

Run the IBM Thin Client for EJB, by completing the following steps.

Procedure

1. Invoke the client application. Run the following Java command:

   Add the following system properties to the Java command if you want authentication and SSL enabled:
2. Provide IIOP authentication configuration and Client SSL Configuration. Add the following system properties to the Java command:

   ```
   -Dcom.ibm.SSL.ConfigURL=file:///home/user1/ssl.client.props
   -Dcom.ibm.CORBA.ConfigURL=file:///home/user1/sas.client.props
   ```

   You can obtain the `ssl.client.props` file and `sas.client.props` file from the WebSphere Application Server installation and modify the file to suit your environment. You must, at a minimum, update the location of the key files in the `ssl.client.props` file to the match location of your target environment. For example,

   ```
   -Dcom.ibm.ssl.keyStore=/home/user1/etc/key.p12
   -Dcom.ibm.ssl.trustStore=/home/user1/etc/trust.p12
   ```

   Recommended SSL configuration settings when running the application with a non-IBM product JRE:

   ```
   com.ibm.ssl.protocol=SSL
   com.ibm.ssl.trustManager=SunX509
   com.ibm.ssl.keyManager=SunX509
   com.ibm.ssl.contextProvider=SunJSSE

   com.ibm.ssl.keyStoreType=JKS
   com.ibm.ssl.keyStoreProvider=SUN
   com.ibm.ssl.keyStore=/home/user1/etc/key.jks

   com.ibm.ssl.trustStoreType=JKS
   com.ibm.ssl.trustStoreProvider=SUN
   com.ibm.ssl.trustStore=/home/user1/etc/trust.jks
   ```

   The key store file and trust store file must be created using the Java keytool utility before the application runs. The automatic key file generation is not supported with a non-IBM product JRE.

   You must override the default ORB implementation of the non-IBM product JRE with the `com.ibm.ws.orb_8.0.0.jar` file, or add it to the classpath.
3. Run your client application:
   - Enter the following command if you have copied the endorsed_apis_8.0.0.jar file into the `JAVA_JRE\lib\endorsed` default directory; for example:

```
$JAVA_HOME/bin/java -Dcom.ibm.SSL.ConfigURL=file:///home/sample/ssl.client.props <your_client_application>
```

- Enter the following command if you have copied the endorsed_apis_8.0.0.jar file into a directory other than the default JAVA_JRE\lib\endorsed directory; for example:

```
$JAVA_HOME/bin/java
-Djava.endorsed.dirs=<directory_that_includes_endorsed_apis_8.0.0.jar>
-Dcom.ibm.SSL.ConfigURL=file:///home/sample/ssl.client.props <your_client_application>
```

What to do next

Enable trace for the IBM Thin Client for EJB by adding the following to the Java command.

```
-Dcom.ibm.ejs.ras.lite.traceSpecification=*=all
```

Running Java thin client applications

You can run Java thin client applications on machines installed with either a WebSphere Application Client installation or a WebSphere Application Server installation.

About this task

Important: Java thin clients are not packaged with JDBC provider classes. For example, the WebSphere Application Server Version 7.0 Java thin client is not packaged with Apache Derby 10.2 classes. Likewise, the version 6.1 Java thin client is not packaged with Cloudscape Version 5.1, Cloudscape Version 10.0, or Cloudscape version 10.1 classes. Therefore, to utilize the JDBC provider classes (such as Apache Derby, Oracle, DB2, Informix®, or Sybase) on a Java thin client, you must:

1. Add the classes to your Java thin client application environment.
2. Make the classes visible to the Java thin client application. To do this, add the path to the classes in the client classpath within the script that launched the client program.

Otherwise, any attempt to load a database class (such as through the JNDI lookup of a datasource) results in a ClassNotFoundException.

The Java invocation to run a Java thin client application varies between a client and a server. If your Java thin client application needs to run on both a client installation and a server installation, follow the steps in the Running a Java thin client application on a server machine topic.

Procedure

- “Running a Java thin client application on a client machine” on page 159
- “Running a Java thin client application on a server machine” on page 159

Example

Your Java thin application client no longer needs additional code to set security providers if you have enabled security for your WebSphere Application Server instance. This code found in IBM i Java thin or pluggable application clients should be removed to prevent migration and compatibility problems. The java.security file from your WebSphere instance in the properties directory is now used to configure the security providers.

- Running the thin or pluggable application client with security enabled

Running the thin or pluggable application client with security enabled. The following code examples illustrates how security providers were set programmatically in the main() method and occurred prior to any code that accessed enterprise beans:

```
import java.security.*;
  ...
  if (System.getProperty("os.name").equals("OS/400")) {

    // Set the default provider list first.
```

```
   Provider jceProv = null;
   Provider jsseProv = null;
   Provider sunProv = null;

   // Allow for when the Provider is not needed, when
   // it is not in the client application's classpath.
   try {
     jceProv = new com.ibm.crypto.provider.IBMJCE();
   }
   catch (Exception ex) {
ex.printStackTrace();
   throw new Exception("Unable to acquire provider.");
}

   try {
     jsseProv = new com.ibm.jsse.JSSEProvider();
   }
   catch (Exception ex) {
ex.printStackTrace();
   throw new Exception("Unable to acquire provider.");
}

   try {
     sunProv = new sun.security.provider.Sun();
   }
   catch (Exception ex) {
ex.printStackTrace();
   throw new Exception("Unable to acquire provider.");
}

   // Enable providers early and ahead of other providers
   // for consistent performance and function.
   if ( (null != sunProv) && (1 != Security.insertProviderAt(sunProv, 1)) ) {
     Security.removeProvider(sunProv.getName());
     Security.insertProviderAt(sunProv, 1);
   }
   if ( (null != jceProv) && (2 != Security.insertProviderAt(jceProv, 2)) ) {
     Security.removeProvider(jceProv.getName());
     Security.insertProviderAt(jceProv, 2);
   }
   if ( (null != jsseProv) && (3 != Security.insertProviderAt(jsseProv, 3)) ) {
     Security.removeProvider(jsseProv.getName());
     Security.insertProviderAt(jsseProv, 3);
   }

   // Adjust default ordering based on admin/startstd properties file.
   // Maximum allowed in property file is 20.
   String provName;
   Class  provClass;
   Object provObj =  null;

   for (int i = 0; i < 21; i++) {
     provName = System.getProperty("os400.security.provider."+ i);

     if (null != provName) {

       try {
         provClass = Class.forName(provName);
         provObj = provClass.newInstance();
       }
       catch (Exception ex) {
         // provider not found
         continue;
       }

       if (i != Security.insertProviderAt((Provider) provObj, i)) {
```

```
          // index 0 adds to end of existing list
          if (i != 0) {
            Security.removeProvider(((Provider) provObj).getName());
            Security.insertProviderAt((Provider) provObj, i);
          }
        }
      } // end if (null != provName)
    } // end for (int i = 0; i < 21; i++)
  } // end if ("os.name").equals("OS/400")
```

For examples of Java thin client applications, refer to the Samples section of the information center.

Running a Java thin client application on a client machine

To run a Java thin client application on a machine with Application Client for WebSphere Application Server installed, use the setup Client command then start the application.

Before you begin

Before performing this task, you must install the Java thin application client from the Application Client for WebSphere Application Server installation.

Procedure

1. Set up the client application environment. Run the setupClient command.

   Use the setupClient script.
   a. Start the Qshell environment. On the CL command line, run the `STRQSH` command.
   b. On the Qshell command line, run the following command using the dot (.) operator:

      `. app_client_root/bin/setupClient [-profileName profileName]`

2. Run a Java command to invoke your client application.

   Run the following command on the Qshell command line:

```
java ${JAVA_FLAGS_EXT} -classpath "$WAS_CLASSPATH:jars_and_classes" -Djava.naming.provider.url=URL class_name app_parm
```

Running a Java thin client application on a server machine

To run a Java thin client application on a machine with WebSphere Application Server installed, use the setupClient command then start the application.

Before you begin

You must install WebSphere Application Server before performing this task.

Procedure

1. Set up the Thin application client environment.

   Use the setupClient script.
   a. Start the Qshell environment. On the CL command line, run the STRQSH command.
   b. On the Qshell command line, run the following command using the dot (.) operator:

      `. app_server_root/bin/setupClient [-profileName profileName]`

2. Run the application client.

   Run the following command on a Qshell command line.

```
java ${JAVA_FLAGS_EXT} -classpath "$WAS_CLASSPATH:jars_and_classes" -Djava.naming.provider.url=URL class_name app_parm
```

   When using the WebSphere Application Server launcher, run the following command on a Qshell command line:

```
java ${JAVA_FLAGS_EXT} -classpath "$WAS_CLASSPATH:jars_and_classes" -Djava.naming.provider.url=URL com.ibm.ws.bootstrap.
```

Managing resources for Java EE client applications

You can manage resources for Java Platform, Enterprise Edition (Java EE) application clients by using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

Before you can manage resources for a Java EE client application, you must have deployed that application.

About this task

After deploying a Java EE client application client, you might want to or need to update the resources that you configured for that client application.

If you want to manage the resources of a Java EE client application deployed on z/OS, you can run the Application Client Resource Configuration Tool (ACRCT) on Windows, according to the following steps, and then reinstall the application on z/OS.

Updating data source and data source provider configurations with the Application Client Resource Configuration Tool

You can update the configuration of an existing data source or data source provider using the Application Client Resource Configuration Tool (ACRCT).

About this task

During this task, you update the configuration of an existing data source or data source provider. Perform this task when your database configuration changes.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT), and open the Enterprise Archive (EAR) file containing the data source or data source provider. The EAR file contents display in a tree view.
2. Select Java Archive (JAR) file from the navigation tree containing the data source or data source provider to update.
3. Expand the JAR file to view its contents until you locate the particular data source or data source provider to update. Take one of the following actions:
   - Right-click the data source object and click **Properties**.
   - Click **Edit > Properties** on the menu bar.
4. Update the properties in the displayed fields. For detailed field help, refer to the Data source provider properties topic.
5. Click **OK** when you finish.
6. Click **File > Save** on the menu bar to save your changes.

Updating URLs and URL provider configurations for application clients

You can update URLs and URL provider configurations for application clients using the Application Client Resource Configuration Tool (ACRCT).

Procedure

1. Start the tool and open the Enterprise Archive (EAR) file containing the URL or URL provider. The EAR file contents are displayed in a tree view.
2. Select from the tree the Java Archive (JAR) file containing the URL or URL provider to update.

3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular URL or URL provider to update. Take one of the following actions:
   a. Right-click the URL object and click **Properties**.
   b. Click **Edit** > **Properties** on the menu bar.
5. Update the properties in the displayed fields.
6. Click **OK** when you finish.
7. Click **File > Save** on the menu bar to save your changes.

Updating mail session configurations for application clients

You can update the configuration of an existing JavaMail session using the Application Client Resource Configuration Tool (ACRCT).

About this task

During this task, you update the configuration of an existing JavaMail session. You cannot update the name of the default JavaMail provider, and you cannot delete the default JavaMail provider from the navigation tree.

Procedure

1. Start the tool and open the Enterprise Archive (EAR) file containing the JavaMail session. The EAR file contents are displayed in the navigation tree view.
2. Select the Java Archive (JAR) file containing the JavaMail session to update from the navigation tree.
3. Expand the JAR file to view its contents.
4. Keep expanding the JAR file contents until you locate the particular JavaMail session to update. Take one of the following actions:
   a. Right-click the object and click **Properties**
   b. Click **Edit > Properties** from the menu bar.
5. Update the properties in the displayed fields.
6. Click **OK** when you finish.
7. Select **File > Save** from the menu bar to save your changes.

Updating Java Message Service provider, connection factories, and destination configurations for application clients

You can update the configuration of an existing Java Message Service (JMS) provider, connection factory or destination using the Application Client Resource Configuration Tool (ACRCT).

About this task

During this task, you update the configuration of an existing Java Message Service (JMS) provider, connection factory or destination.

Procedure

1. Start the tool and open the Enterprise Archive (EAR) file containing the Java Message Service (JMS) provider, connection factory, or destination. The EAR file contents display in a tree view.
2. Select the Java Archive (JAR) file containing the JMS provider, connection factory, or destination to update from the navigation tree.
3. Expand the JAR file to view its contents until you locate the particular JMS provider, connection factory, or destination to update. When you find it, do one of the following actions:
   - Right-click the provider, and click **Properties**.

- Click **Edit > Properties** on the menu bar.

4. Update the properties in the displayed fields. For detailed field help, see:
   - JMS provider properties
   - WebSphere Application Server Queue connection factory properties
   - WebSphere Application Server Topic connection factory properties
   - WebSphere Application Server Queue destination properties
   - WebSphere Application Server Topic destination properties
5. Click **OK**.
6. Click **File > Save** to save your changes.

Updating WebSphere MQ as a Java Message Service provider, and its JMS resource configurations, for application clients

You can update an existing configuration of WebSphere MQ as a Java Message Service (JMS) provider, and update the configuration of WebSphere MQ connection factories or WebSphere MQ destinations.

About this task

Use this task to update an existing configuration of WebSphere MQ as a Java Message Service (JMS) provider, and to update the configuration of WebSphere MQ connection factories or WebSphere MQ destinations.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT).
2. Open the Enterprise Archive (EAR) file containing the WebSphere MQ JMS provider, WebSphere MQ connection factory, or WebSphere MQ destination. The EAR file contents are displayed in the navigation tree view.
3. Select the Java Archive (JAR) file containing the JMS provider, connection factory, or destination to update.
4. Expand the JAR file to view its contents until you locate the particular JMS provider, connection factory, or destination that you want to update. Complete one of the following actions:
   - Right-click the appropriate object and click **Properties**.
   - Click **Edit > Properties** on the menu bar.
5. Update the properties in the displayed fields. For detailed field help, see:
   - JMS provider properties
   - MQ Queue connection factory properties
   - MQ Topic connection factory properties
   - MQ Queue destination properties
   - MQ Topic destination properties
6. Click **OK**.
7. Click **File > Save** to save your changes.

Updating resource environment entry and resource environment provider configurations for application clients

You can update the configuration of an existing resource environment entry or resource environment provider using the Application Client Resource Configuration Tool (ACRCT).

About this task

During this task, you update the configuration of an existing resource environment entry or resource environment provider.

Procedure

1. Start the tool and open the Enterprise Archive (EAR) file containing the resource environment entry or resource environment provider. The EAR file contents display in a navigation tree view.
2. Select from the tree the Java Archive (JAR) file containing the resource environment entry or resource environment provider to update.
3. Expand the JAR file to view its contents until you locate the resource environment entry or resource environment provider to update. Take one of the following actions:
   - Right-click the resource environment object, and click **Properties**.
   - Click **Edit > Properties** on the menu bar.
4. Update the properties in the displayed fields. For detailed field help, see:
   - Resource environment provider properties
   - Resource environment entry properties
5. Click **OK** when you finish.
6. Click **File > Save** on the menu bar to save your changes.

Example

- Configuring resource environment custom settings for application clients
- Configuring Resource Environment settings

Configuring resource environment custom settings for application clients: This code example illustrates how the custom page applies to every resource type. You can specify as many custom names and values as you need:

```
<propertySet xmi:id="J2EEResourcePropertySet_20">
<resourceProperties xmi:id="J2EEResourceProperty_22"
name="resourceEnvEntry:customName" value="resourceEnvEntry:customValue"/>
</propertySet>
```

Configuring Resource Environment settings: This code example illustrates how to configure Resource Environment settings:

```
<resources.env:ResourceEnvironmentProvider xmi:id="ResourceEnvironmentProvider_1"
name="resourceEnvProvider:name" description="resourceEnvProvider:description">
<classpath>resourceEnvProvider:classpath</classpath>
<factories xmi:type="resources.env:ResourceEnvEntry" xmi:id="ResourceEnvEntry_1"
name="resourceEnvEntry:name" jndiName="resourceEnvEntry:jndiName"
description="resourceEnvEntry:description">
<propertySet xmi:id="J2EEResourcePropertySet_20">
<resourceProperties xmi:id="J2EEResourceProperty_22"
name="resourceEnvEntry:customName" value="resourceEnvEntry:customValue"/>
</propertySet>
</factories>
<propertySet xmi:id="J2EEResourcePropertySet_21">
<resourceProperties xmi:id="J2EEResourceProperty_23"
name="resourceEnvProvider:customName" value="resourceEnvProvider:customValue"/>
</propertySet>
</resources.env:ResourceEnvironmentProvider>
```

- Required fields:
  - Resource Environment Provider page: Name
  - Resource Environment Entry page: Name, JNDI Name

Removing application client resources

You can remove Java Platform, Enterprise Edition (Java EE) application client resources using the Application Client Resource Configuration Tool (ACRCT).

Before you begin

The option to delete an item does not offer a confirmation dialog. As a safeguard, consider saving your work right before you begin this task. If you change your mind after removing an item, you can close the EAR file without saving your changes, canceling your deletion. Remember to close the EAR file immediately after the deletion, or you also lose any unsaved work that you performed since the deletion.

This task only applies to Java EE application clients.

Procedure

1. Start the Application Client Resource Configuration Tool (ACRCT) and open the Enterprise Archive (EAR) file from which you want to remove an object. The EAR file contents display in the navigation tree view. If you already have an EAR file open and have made some changes, click **File > Save** to save your work before preceding to delete an object.
2. Locate the object that you want to remove in the tree.
3. Right-click the object, and click **Delete**.
4. Click **File > Save**.

clientUpgrade script

The clientUpgrade script migrates application client modules and their resources in an enterprise archive (EAR) file so that these application clients can run in WebSphere Application Server Version 8. The script converts an EAR file that you want to migrate and then overwrites the original EAR file with the converted EAR file.

The following title provides information about the clientUpgrade script.

Attention: This command was deprecated in Version 6.1.

Type	Description
Product	The clientUpgrade script is available in the WebSphere Application Server (WebSphere Application Server, Express and WebSphere Application Server (base)) product only.
Authority	To run this script, your user profile must have *ALLOBJ authority.

Syntax	The syntax of the clientUpgrade script is:
	`clientUpgrade` *`EAR_file`* `[ -clientJAR` *`client_JAR_file`* `]`
	`[ -logFileLocation` *`logFileLocation`* `]`
	`[ -traceString` *`trace_spec`* `[ -traceFile` *`file_name`* `] ]`

Parameters	The parameters of the clientUpgrade script are: • *EAR_file* -- This is a required parameter. The value EAR_file specifies the fully-qualified path of the EAR file that contains the application client modules that you want to migrate. • -clientJAR -- This is an optional parameter. The value client_JAR_file specifies a JAR file that you want to migrate. The script overwrites the original EAR file with a new EAR file that contains only the specified JAR files. If you do not specify this parameter, the clientUpgrade script migrates all client JAR files in the EAR file. • -logFileLocation -- Use this optional parameter to specify an alternate location to store the log output. • -traceString -- This is an optional parameter. The value trace_spec specifies the trace information that you want to collect. To gather all trace information, specify "*=all=enabled" (including the double quotation marks (")). By default, the script does not gather trace information. If you specify this parameter, you must also specify the -traceFile parameter. • -traceFile -- This is an optional parameter. The value file_name The value file_name specifies the name of the output file for trace information. If you specify the -traceString parameter but do not specify the -traceFile parameter, the script does not generate a trace file.

Type	Description
Logging	The clientUpgrade script displays status while it runs. It also saves more extensive logging information to the `clientupgrade.log` file. This file is located in the `/QIBM/UserData/WebSphere/AppServer/V8/edition/profiles/default/logs` directory (for a default installation using the default profile) or in the location specified by the -logFileLocation parameter.

These examples demonstrate correct syntax. In this example, the `My51Application.ear` file is migrated from WebSphere Application Server Version 5.1, The script overwrites the original EAR file with a new file that you can deploy in your WebSphere Application Server Version 8 profile.

```
clientUpgrade /My51Application/My51Application.ear
```

In this example, only the `myJarFile.jar` client JAR file is migrated. The script overwrites `My51Application.ear` with an EAR file that contains `myJarFile.jar`. You can deploy the new EAR file in your WebSphere Application Server profile.

```
clientUpgrade /My51Application/My51Application.ear -clientJAR myJarFile.jar
```

Chapter 6. Administering Communications Enabled Applications

Communications Enabled Applications (CEA) is a functionality that provides the ability to add dynamic web communications to any application or business process. The product provides a suite of integrated telephony and collaborative web services that extends the interactivity of enterprise and web commerce applications. With the CEA capability, enterprise solution architects and developers can use a single core application to enable multiple modes of communication. Enterprise developers do not need to have extensive knowledge of telephony or Session Initiation Protocol (SIP) to implement CEA. The CEA capability delivers call control, notifications, and interactivity and provides the platform for more complex communications.

Administering communications enabled applications

Configuring services for communications enabled applications

CEA settings

Use this page to configure the Representational State Transfer (REST) interface and the computer-telephony integration (CTI) gateway to enable Communications Enabled Applications (CEA). A CTI application manages the event flow that is generated by the telephony switch, IP PBX, during the life cycle of a call.

- To view this administrative console page in a single-server environment, click **Servers > Server Types > WebSphere application servers > *server_name* > Communications Enabled Applications (CEA)**.
- To view this administrative console page in a clustered environment, click **Servers > Clusters > WebSphere application server clusters > *server_cluster* > Communications Enabled Applications (CEA)**.

  Because servers in a cluster are clones of each other, you only need to make configuration changes from the main cluster panel for the cluster, not for each individual server in the cluster.

  The CEA system application uses a SIP session for state replication and therefore relies on replication domains and clustering to provide failover support, similar to any other SIP application. See the topic Replicating SIP sessions for more information.

Enable communications service:

Specifies to enable or disable the communications service for this server or cluster. Disabling the field prevents the service from starting and saves system resources.

Context root:

Specifies the context root of the REST interface. Use this field to assign a different context root to the REST interface.

The context root is combined with the defined servlet mapping for the REST interface to compose the full URL that users type to make a REST request. For example, if the context root is `/gettingstarted` and the servlet mapping is `CommServlet/call`, then the URL is `http://host:port/gettingstarted/CommServlet/call`.

Virtual host:

Specifies the name of the virtual host to which the REST interface is currently mapped.

Expanding the menu list displays a list of the defined virtual hosts. To change a mapping, select a different virtual host from the list.

Maximum hold time:

Specifies the time in seconds in which a GET /event call to the REST interface waits for new or changed data or status before timing out.

Data Type	Integer
Default	30

Use SIP CTI (ECMA TR/87) gateway for telephony access:

Select this option to use the SIP CTI gateway for telephony access.

Host name or IP address:

Specifies the address or fully qualified domain name, FQDM, of the CTI gateway to be connected to by the CEA service.

Data Type	string
Default	localhost

Port:

Specifies the port of the CTI gateway to be connected to by the CEA service.

Data Type	integer
Default	5060

Protocol:

Specifies the protocol to be used when connecting to the CTI, TR/87, gateway. The default value is TCP.

Extract user name from request:

When enabled, an attempt is made to extract the user name from the HTTP request. If the name cannot be extracted, the Superuser name is used. This name is used when opening a new TR/87 session to the CTI gateway.

Attention: After you enable the **Extract user name from request** option, an attempt is made to extract the user name from the HTTP request. However, the user name has a null value even for authenticated users. The CEA Rest Service Servlet is an unprotected URI. Thus, you also must enable the **Use available authentication data when an unprotected URI is accessed** option. Use the administrative console to enable the **Use available authentication data when an unprotected URI is accessed** option under **Security > Global security > Web and SIP security > General settings**. After you enable this option, the user name is available to the CEA Rest Service Servlet during the request.

Superuser name:

Specifies the name that is used when opening a new TR/87 session to the configured CTI gateway. This requires that the CTI gateway be configured with a superuser account that is used to create phone calls on behalf of all end users.

Data Type	string
Default	ceauser

Attention: You can specify a **Superuser name** on the CEA settings panel. The superuser corresponds to a user, who is configured on the PBX, and has the ability to control any phone that is configured on the PBX. Also, the superuser has the ability to control multiple phones concurrently. This functionality, however, is not supported on every PBX. The Cisco PBX requires that you set a user name for each phone that you want to control. The **Superuser name** field on the CEA settings panel can only pass a single user name; therefore, it can only control a single device. To control multiple phones concurrently using the Cisco PBX, you must derive the user name from the user credentials for this PBX. To accomplish this task, ensure that the **Extract user name from request** check box is selected on the CEA settings panel.

Use a third-party Web services provider for telephony access:

Select this option to specify a third-party Web services provider for telephony access. Instead of using SIP CTI, this approach uses a third-party service that has implemented specific Web services to utilize a different method to connect to the telephony infrastructure.

Third-party Web services provider's WSDL:

The URL path pointing to a third-party Web services provider's WSDL. If a value is specified, the SIP CTI gateway is not used.

CEA custom properties

This topic discusses the CEA custom properties that you can set on the administrative console.

To view the administrative console page associated with this topic, click **Servers > Clusters > WebSphere application server clusters > *server_cluster* > Communications Enabled Applications (CEA) > Custom properties**.

Specify a property and its value as a name-value pair on the Custom properties page. You can use the custom properties page to define the following CEA custom properties:

- “SIP_RFC3263_auto_resolve”
- “sipOverTlsPbxOverride”

SIP_RFC3263_auto_resolve:

Disables the automatic DNS resolve for SIP messages. If this property is set to `false` for a SIP container, DNS resolve is only triggered by application APIs.

Data type	Boolean
Default	True

sipOverTlsPbxOverride:

Changes proxy behavior to send "sip:" formatted messages over a TLS connection instead of the default "sips:" message format.

Data type	String
Default	True

Configuring communications enabled applications in a cluster

You can configure multiple application servers to use the Communications Enabled Applications (CEA) capability in a cluster environment to balance workload demands.

About this task

To scale to two or more application servers running CEA, you must create a cluster that includes each application server. Clusters are groups of servers that are managed together and participate in workload management. You must use proxy servers to distribute the requests to the CEA application servers, but not IBM HTTP Servers. CEA is implemented as a converged application that combines both HTTP and session initiation protocol (SIP) protocols. However, IBM HTTP Server is limited to handling the HTTP protocol. The proxy server is a converged proxy so it handles both HTTP and SIP protocols.

Procedure

1. Identify the application servers or nodes that are running CEA that you want to manage as a cluster.
2. Create the cluster using the administrative console. For details on creating the cluster, read about creating a cluster using basic cluster settings.
3. Install a SIP proxy server. The SIP proxy server routes HTTP and SIP requests to the back-end application servers. This proxy server provides high performance SIP proxy capabilities that you can use at the edge of the network to route, load balance, and improve response times for SIP dialogs to backend SIP resources. For details on installing a SIP proxy server, read about installing a SIP proxy server.
4. Configure the SIP converged proxy server. Select the default cluster to specify where you want the SIP proxy to route requests. After you choose a default cluster, your SIP proxy server is functional.

Results

You have successfully configured a cluster environment for CEA applications.

Chapter 7. Administering Data access resources

This page provides a starting point for finding information about data access. Various enterprise information systems (EIS) use different methods for storing data. These backend data stores might be relational databases, procedural transaction programs, or object-oriented databases.

The flexible IBM WebSphere Application Server provides several options for accessing an information system backend data store:

- Programming directly to the database through the JDBC 4.0 API, JDBC 3.0 API, or JDBC 2.0 optional package API.
- Programming to the procedural backend transaction through various J2EE Connector Architecture (JCA) 1.0 or 1.5 compliant connectors.
- Programming in the bean-managed persistence (BMP) bean or servlets indirectly accessing the backend store through either the JDBC API or JCA-compliant connectors.
- Using container-managed persistence (CMP) beans.
- Using the IBM data access beans, which also use the JDBC API, but give you a rich set of features and function that hide much of the complexity associated with accessing relational databases.

Service Data Objects (SDO) simplify the programmer experience with a universal abstraction for messages and data, whether the programmer thinks of data in terms of XML documents or Java objects. For programmers, SDOs eliminate the complexity of the underlying data access technology such as, JDBC, RMI/IIOP, JAX-RPC, and JMS, and message transport technology such as, java.io.Serializable, DOM Objects, SOAP, and JMS.

Deploying data access applications

Deploying a data access application includes more than installing your web application archive (WAR) or enterprise archive (EAR) file onto a server. Deployment can include tasks for configuring your application to use the data access resources of the server and overall runtime environment.

Before you begin

You can deploy only application code that is assembled into the appropriate modules. See the topic, Assembling data access applications for guidelines, for this process.

About this task

Perform the following steps if your application requires access to a relational database (RDB). When your application requires access to a different type of enterprise information system (EIS), such as an object-oriented database or the Customer Information Control System (CICS®), consult the topics, Relational resource adapters and JCA, and Accessing data using Java EE Connector Architecture connectors.

Procedure

1. If your RDB configuration does not exist, do the following steps:
   a. Create a database to hold the data.
   b. Create tables required by your application.
      If your application uses container managed persistence (CMP) entity beans to access the data You can create the tables using the data definition language (DDL) generated from the enterprise bean configuration. For more information, see the topic, Recreating database tables from the exported table data definition language.

If your application uses bean managed persistence (BMP) entity beans, or *does not* **use entity beans**

You must use your database server interfaces to create the tables.

The Enterprise JavaBeans (EJB) to RDB Mapping wizard of an assembly tool is also used to create your database tables for either type of entity bean. Select the top-down mapping option in the wizard. However, this option does not give you direct control in naming the RDB elements or choosing column types. Additionally, because the top-down process is automatic, it might not provide mappings to reflect the precise relationships that you intend.

If you use Rational Application Developer, consult the information center about the mapping wizard. To learn about all of your assembly tool options, see the assembly tools topic in this information center.

c. Check the data source minimum required settings by vendor to see any database vendor requirements for connecting to an application server. See the topic, Data source minimum required settings, by vendor, for instructions.

2. Optional: Map your entity beans to the database tables through the meet-in-the-middle mapping option of an assembly tool. Complete this step only if you did not create your database schema through the top-down mapping option, did not generate your mapping relationships through bottom-up mapping, or did not generate mappings during the application assembly process. For information about the top-down mapping option see the information center for Rational Application Developer.
3. Install your application onto the application server. See the topic, Installing enterprise application files. When you install the application, you can alter data access settings that were made during application assembly, or, if they were omitted from the assembly process, set them for the first time. These settings include resource bindings and resource authentication aliases, which are addressed in the following substeps:
   a. Bind application resource references to the data sources, or other resource objects, that provide database connectivity. For details on the concept of binding, see the topic, Data source lookups for enterprise beans and web modules.

      Tip: After deployment, you can use the WebSphere Application Server administrative console to alter resource bindings. Click **Applications** > **Application Types** > **Webphere enterprise applications** > ***application_name***, and select the link to the appropriate mapping page. For example, if you want to alter the binding of an EJB module resource, you might click **Map data sources for all 2.x CMP beans**. For a web module resource, click **Resource references**.

   b. Define authentication alias data for resources that must be authenticated with the backend through *container-managed* authorization. In this security configuration, WebSphere Application Server performs EIS signon for data source or connection factory connections. Consult the topic, J2EE connector security for detailed reference on resource authentication.
4. Start the deployed application files using the administrative console, the wsadmin scripting tool startApplication command, or your own Java program.
5. Save the changes to your administrative configuration.
6. Test the application. For example, point a web browser at the URL for a deployed application and examine the performance of the application.

Results

When you deploy an application that uses a DB2 UDB for IBM i back-end database, you might find the following exception in the SystemOut.log file:

```
PMGR6022E: Error using adapter to create or execute an Interaction
```

This type of error indicates that you deployed an application with container-managed persistence (CMP) enterprise beans that were originally configured to access a DB2 database on Windows, Linux, or a supported UNIX system. Using the administrative console, uninstall the affected CMP applications, then

reinstall the applications with the new database setting. Remember to select **Deploy enterprise beans**; on the **EJB deploy** panel, select the appropriate version of your DB2 UDB for IBM i database.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

What to do next

If the application does not perform as wanted, update the application, then save and test it again.

Available resources

Use this page to select configured resources that you want to bind to the resource references of the enterprise beans or web modules in your application.

To view this administrative console page:

1. Click **Applications** > **Application Types** > **Websphere enterprise applications** > ***application_name***.
2. Click the link for any of these resource configuration pages:
   - **Resource references**
   - **Map data sources for all 2.x CMP beans**
   - **Provide default data source mapping for modules containing 2.x entity beans**
3. Locate the table row of the EJB or web module that you want to map to a different resource.
4. Within the row, locate the JNDI name of the resource that is currently bound to the EJB or web module.
5. Click **Browse**.

   You now see **Available resources**.

Each table row corresponds to a resource that you can bind to your enterprise bean or web module.

Select

Select the resource that you want to bind to the resource reference of your module.

JNDI name

The Java Naming and Directory Interface (JNDI) name of the resource that you want to bind to the resource reference of your module.

Data type String

Scope

The scope of the resource. Note that this administrative console page displays only resources that are configured for a scope at which your application operates.

Description

The text description of the resource.

Map data sources for all 1.x CMP beans

Use this page to designate how the container-managed persistence (CMP) 1.x beans of an application map to data sources that are available to the application.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Map data sources for all 1.x CMP beans**.

Guidelines for using this administrative console page:

- The table depicts the 1.x CMP bean contents of your application.
- Each table row corresponds to a CMP bean within a specific EJB module. A row shows the JNDI name of the data source mapping target of the bean *only* if you bound them together during application assembly or installation. For every data source that is displayed, you see the corresponding security configuration.
- To set your mappings:
  1. Select a row. Be aware that if you check multiple rows on this page, the data source mapping target that you select in step 2 applies to all of those CMP beans.
  2. Click **Browse** to select a data source from the new page that is displayed, the Available Resources page. The Available Resources page shows all data sources that are available mapping targets for your CMP beans.
  3. Click **Apply**. The console displays the 1.x CMP bean data sources page again. In the rows that you previously selected, you now see the JNDI name of the new resource mapping target.
  4. *Before* you click **OK** to save your new configuration, set the security parameters for the data source. Use the following steps.
- To specify data source security settings:
  1. Select one or more rows in the table.
  2. Type in a user name and password that comprise the authentication alias for signing on to the data source. If these entries are not listed in the application Java Platform, Enterprise Edition (Java EE) Connector (J2C) authentication data list, you must input them into the list after saving your settings on this page. Read the information center topic on managing Java EE Connector Architecture authentication data entries for more information.
  3. Click **Apply** that immediately follows the user name and password input fields.
- Repeat all of the previous steps as necessary.
- Click **OK** to save your settings.

Select
Select the check boxes of the rows that you want to edit.

EJB
The name of an enterprise bean in the application.

EJB Module
The name of the module that contains the enterprise bean.

URI
Specifies location of the module relative to the root of the application EAR file.

JNDI name
The Java Naming and Directory Interface (JNDI) name of the data source that is configured for the enterprise bean.

Data type String

User name
The user name and password that comprise the authentication alias for securing the data source.

Map default data sources for modules containing 1.x entity beans

Use this page to set the default data source mapping for EJB modules that contain 1.x container-managed persistence (CMP) beans. Unless you configure individual data sources for your 1.x CMP beans, this default mapping applies to all beans within the module.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Map default data sources for modules containing 1.x entity beans**.

Guidelines for using this administrative console page:

- The page displays a table that depicts the EJB modules in your application that contain 1.x CMP beans.
- Each table row corresponds to a module. A row shows the JNDI name of the data source mapping target of the EJB module *only* if you bound them together during application assembly. For every data source that is displayed, you see the corresponding security configuration.
- To set your default data source mappings:
  1. Select a row. Be aware that if you check multiple rows on this page, the data source mapping target that you select in step 2 applies to all of those EJB modules.
  2. Click **Browse** to select a data source from the new page that is displayed, the Available Resources page. The Available Resources page shows all data sources that are available mapping targets for your EJB modules.
  3. Click **Apply**. The console displays the 1.x entity bean data sources page again. In the rows that you previously selected, you now see the JNDI name of the new resource mapping target.
  4. *Before* you click **OK** to save your new configuration, set the security parameters for the data source. Use the following steps.
- To specify security settings for the default data source:
  1. Select a row. Be aware that if you check multiple rows on this page, the security settings that you select later apply to all of those data sources.
  2. Type in a user name and password that comprise the authentication alias for signing on to the data source. If these entries are not listed in the application Java Platform, Enterprise Edition (Java EE) Connector (J2C) authentication data list, you must input them into the list after saving your settings on this page. Read the information center topic on managing Java EE Connector Architecture authentication data entries for more information.
  3. Click **Apply** that immediately follows the user name and password input fields.
- Repeat all of the previous steps as necessary.
- Click **OK** to save your work.

Select

Select the check boxes of the rows that you want to edit.

EJB Module

The name of the module that contains the 1.x enterprise beans.

URI

Specifies location of the module relative to the root of the application EAR file.

JNDI name

The Java Naming and Directory Interface (JNDI) name of the default data source for the EJB module.

Data type String

User name

The user name and password that comprise the authentication alias for securing the data source.

Map data sources for all 2.x CMP beans settings

Use this page to map container-managed persistence (CMP) 2.x beans of an application to data sources that are available to the application.

To view this administrative console page, click **Applications** > **Application Types** > **Websphere enterprise applications** > ***application_name*** > **Map data sources for all 2.x CMP beans**.

Each table row corresponds to a CMP bean within a specific EJB module. A row shows the JNDI name of the data source mapping target of the bean only if you bound them together during application assembly. For every data source that is displayed, you see the corresponding security configuration.

Set Multiple JNDI names

Specify the Java Naming and Directory Interface (JNDI) name for multiple EJB modules. Select one or more EJB modules from the table, and select a JNDI name from this list to configure the EJB modules with that JNDI name.

Data type Drop-down list

Set Authorization Type

Specify the authorization type for securing the data source. Select one or more EJB modules from the table to set the authorization type.

Select either **Container** or **Application** from the displayed list. Container-managed authorization indicates that WebSphere Application Server performs signon to the data source. Application-managed authorization indicates that the enterprise bean code performs signon.

Modify Resource Authentication Method

Specify the authorization type and the authentication method for securing the data source. Select one or more EJB modules from the table to modify the resource authentication method.

You can choose between the following authentication methods:

- **None**:
  1. Determine which data source configurations to designate with no authentication method.
  2. Select the appropriate table rows.
  3. Select **None** from the list of authentication method options that precede the table.
  4. Click **Apply**.
- **Use default method (many-to-one mapping)**:
  1. Determine which data source configurations to designate with the WebSphere Application Server DefaultPrincipalMapping login configuration. Apply this option to each data source individually if you want to designate different authentication data aliases. See the information center topic on J2EE Connector security for more information on the default mapping configuration.
  2. Select the appropriate table rows.
  3. Select **Use default method (many-to-one mapping)** from the list of authentication method options that precede the table.
  4. Select an authentication data entry or alias from the list.
  5. Click **Apply**.
- **Use Kerberos authentication**: Specifies to use the Kerberos authentication method.
  1. Ensure that you have configured the Kerberos authentication mechanism in the application server.
  2. Select the appropriate table row.
  3. Select **Use Kerberos authentication** from the list of authentication method options that precede the table.

4. Select an application login configuration from the list.
5. Click **Apply**.
6. To edit the properties of the custom login configuration, click **Mapping Properties** in the table cell.

The application server will attempt to verify that you are connecting to the correct type of database when you select this option.

- **Use trusted connections (one-to-one mapping)**:
  1. Determine which data source configurations to designate with a custom Java Authentication and Authorization Service (JAAS) login configuration. See the information center topic on J2EE Connector security for more information on custom JAAS login configurations.
  2. Select the appropriate table row.
  3. Ensure that the database to which the modules will connect is configured for trusted connections.
  4. Select **Use trusted connections (one-to-one mapping)** from the list of authentication method options that precede the table.
  5. Select an application login configuration from the list.
  6. Click **Apply**.

  The application server will attempt to verify that you are connecting to the correct type of database when you select this option.
- **Custom login configuration**:
  1. Determine which data source configurations to designate with a custom Java Authentication and Authorization Service (JAAS) login configuration. See the information center topic on J2EE Connector security for more information on custom JAAS login configurations.
  2. Select the appropriate table row.
  3. Select **Use custom login configuration** from the list of authentication method options that precede the table.
  4. Select an application login configuration from the list.
  5. Click **Apply**.
  6. To edit the properties of the custom login configuration, click **Mapping Properties** in the table cell.

Select
Select the check boxes of the rows that you want to edit.

EJB
The name of an enterprise bean in the application.

EJB Module
The name of the module that contains the enterprise bean.

URI
Specifies location of the module relative to the root of the application EAR file.

Target resource JNDI name
Specifies the resource to which the CMP bean is bound.

Resource authorization
Specifies the current setting for the resource authorization type.

Modify this setting with **Set authorization type**.

Map data sources for all 2.x CMP beans

Use this page to set the default data source mapping for EJB modules that contain 2.x container-managed persistence (CMP) beans. Unless you configure individual data sources for your 2.x CMP beans, this default mapping applies to all beans within the module.

To view this administrative console panel, click **Applications** > **Application Types** > **Websphere enterprise applications** > ***application_name*** > **Map data sources for all 2.x CMP beans** .

This panel displays a table that depicts the EJB modules in your application that contain 2.x CMP beans. Each table row corresponds to a module. A row shows the JNDI name of the data source mapping target of the EJB module only if you bound them together during application assembly. For every data source that is displayed, you see the corresponding security configuration.

Set Multiple JNDI Names

Specifies the JNDI name to bind to one or more modules. Select one or more modules, click **Set Multiple JNDI Names**, and select the JNDI name for the resource to which you would like to bind the module.

Set Authorization Type

Specifies the authorization type that you to use for the modules. Select one or more modules, click **Set Authorization Type**, and select the authorization type.

You can choose:

- Per application - indicates that the enterprise bean code performs signon.
- Container - indicates that the application server performs signon to the data source.

Modify Resource Authentication Method

Specifies the resource authentication method for the modules that you have configured with container-managed authorization. Select one or more modules, click **Modify Resource Authentication Method**, and select the authentication method.

You can choose between the following authentication methods:

- **None**:
  1. Determine which data source configurations to designate with no authentication method.
  2. Select the appropriate table rows.
  3. Select **None** from the list of authentication method options that precede the table.
  4. Click **Apply**.
- **Use default method (many-to-one mapping)**:
  1. Determine which data source configurations to designate with the WebSphere Application Server DefaultPrincipalMapping login configuration. Apply this option to each data source individually if you want to designate different authentication data aliases. See the information center topic on J2EE Connector security for more information on the default mapping configuration.
  2. Select the appropriate table rows.
  3. Select **Use default method (many-to-one mapping)** from the list of authentication method options that precede the table.
  4. Select an authentication data entry or alias from the list.
  5. Click **Apply**.
- **Use Kerberos authentication**: Specifies to use the Kerberos authentication method.
  1. Ensure that you have configured the Kerberos authentication mechanism in the application server.
  2. Select the appropriate table row.
  3. Select **Use Kerberos authentication** from the list of authentication method options that precede the table.

4. Select an application login configuration from the list.
5. Click **Apply**.
6. To edit the properties of the custom login configuration, click **Mapping Properties** in the table cell.

The application server will attempt to verify that you are connecting to the correct type of database when you select this option.

- **Use trusted connections (one-to-one mapping)**:
  1. Determine which data source configurations to designate with a custom Java Authentication and Authorization Service (JAAS) login configuration. See the information center topic on J2EE Connector security for more information on custom JAAS login configurations.
  2. Select the appropriate table row.
  3. Ensure that the database to which the modules will connect is configured for trusted connections.
  4. Select **Use trusted connections (one-to-one mapping)** from the list of authentication method options that precede the table.
  5. Select an application login configuration from the list.
  6. Click **Apply**.

  The application server will attempt to verify that you are connecting to the correct type of database when you select this option.
- **Custom login configuration**:
  1. Determine which data source configurations to designate with a custom Java Authentication and Authorization Service (JAAS) login configuration. See the information center topic on J2EE Connector security for more information on custom JAAS login configurations.
  2. Select the appropriate table row.
  3. Select **Use custom login configuration** from the list of authentication method options that precede the table.
  4. Select an application login configuration from the list.
  5. Click **Apply**.
  6. To edit the properties of the custom login configuration, click **Mapping Properties** in the table cell.

Select
Select the check boxes of the rows you want to edit.

EJB Module
Specifies the name of the module that contains the 2.x enterprise beans.

URI
Specifies location of the module relative to the root of the application EAR file.

JNDI name
Specifies the Java Naming and Directory Interface (JNDI) name of the default data source for the EJB module.

Data type String

Resource authorization
Specifies the authorization type and the authentication method for securing the data source.

Extended Datasource Properties
When selected, you will be directed to a panel on which you can specify extended properties that the module can use for the DB2 data source.

The application server will attempt to verify that you are connecting to the correct type of database when you select this option.

Installing a resource adapter archive

The application server uses the classes and other code that comprise a resource adapter archive (RAR) to support the resource adapters that you configure.

Before you begin

A RAR file, which is often called a Java EE Connector Architecture (JCA) connector, must comply with the JCA Specification. You can meet these requirements by using a supported assembly tool to assemble a collection of Java archive (JAR) files, other runnable components, and utility classes into a deployable resource adapter archive (RAR). You can then install the RAR file in the application server.

About this task

A resource adapter archive provides the classes and other code to support a resource adapter for access to a specific EIS, such as the Customer Information Control System (CICS). Therefore, you can only configure resource adapters for an EIS after you install the appropriate RAR file.

Important: When you use the **Install RAR** dialog to install a RAR file, the scope you define on the Resource Adapters page has no effect on where the RAR file is installed. You can install RAR files only at the node level, which you specify on the Install RAR page. To set the scope of an RAR file to a specific cluster, or server, after you install the RAR file at each node level, create a copy of the RAR file with the appropriate cluster or server scope.

Procedure

1. Navigate to the **Resource adapter** panel. Click **Resources** > **Resource Adapters** > **Resource adapters**.
2. Install a new resource adapter archive.
   a. Click **Install RAR**. A dialog opens for installing a RAR file and configuring the associated resource adapter. Only click **New** if you want to configure a new resource adapter for a previously installed RAR file.
   b. Browse to find the appropriate RAR file.
      - If your RAR file is located on your local workstation, select **Local path**, and browse to find the file.
      - If your RAR file is located on your server, select **Remote file system**, and specify the fully qualified path to the file.
   c. Click **Next**.
3. Configure the resource adapter name and any other properties needed under *General Properties*. For more details on the settings that you can configure, such as the J2C connection factories, see the topics Installing resource adapters within applications and Configuring resource adapters.
4. Click **OK**.
5. Optional: Create a copy of the RAR file with a different scope level. After you install the RAR file at each node level, you can create another copy of the file that has a specific server or cluster as the scope for that file.

   Note:

   - If you do not create a copy of your RAR at the cluster scope, then you must create identical factories (connection factories, admin object, and activation specifications) at the node level for each of your nodes in the cluster. By creating the copy of your RAR, you provide a

placeholder for your factories and circumvent the need to create identical factories at the node level for each of your nodes in the cluster.

- You must still install the RAR binaries (files, such as jars and xml deployment files) on each node for the RAR to operate successfully.

a. Click **Resources**.
b. Click **Resource Adapters**.
c. Select the scope level and then click **NEW**.
d. Choose the RAR file from the installed archive path.
e. Click **OK**.

Results

You have installed a resource adapter archive that provide access to the EIS when it is properly configured. If you must configure more settings, or change some settings that were configured during the installation process, refer to the topic on configuring a resource adapter in the administrative console for more information.

Installing resource adapters embedded within applications

Install resource adapters in your applications so they can access outside data sources.

Before you begin

The JCA Version 1.6 specification adds support for Java annotations in RAR modules. For more information on annotation support see the topic, JCA 1.6 support for annotations in RAR modules.

About this task

Procedure

1. Assemble an application with RAR modules in it. See the topic Assembling applications for more information.
2. Install the application. Follow the steps in the topic Installing a new application.

   In the **Map modules to servers** step, specify target servers or clusters for each RAR file. Be sure to map all other modules that use the resource adapters defined in the RAR modules to the same targets. Also, specify the web servers as targets that serve as routers for requests to this application. The plug-in configuration file (`plugin-cfg.xml`) for each web server is generated based on the applications that are routed through it.

   In the **Metadata for modules** step of installing an application, you can set or unset the metadata-complete flag as discussed in the topic, JCA 1.6 support for annotations in RAR modules.

   Note: When installing a RAR file on a server, the application server looks for the manifest (MANIFEST.MF) for the connector module. The application server first looks for the RAR file's `connectorModule.jar` file and loads the manifest from the `connectorModule.jar` file. If the class path entry is in the manifest from the `connectorModule.jar`file, the RAR uses that class path.

   To ensure that the installed connector module finds the classes and resources that it needs, check the Class path setting for the RAR using the administrative console. For more information on how to check this setting, see the topics Resource adapter settings and WebSphere relational resource adapter settings.
3. Click **Finish** > **Save** to save the changes.
4. Create connection factories for the newly installed application.

   See the topic, Configuring connection factories for resource adapters within applications to view the steps to complete this step.

Results

Note: A given native library can only be loaded one time for each instance of the Java virtual machine (JVM). Because each application has its own class loader, separate applications with embedded RAR files cannot both use the same native library. The second application receives an exception when it tries to load the library.

If any application deployed on the application server uses an embedded RAR file that includes native path elements, then you must always ensure that you shut down the application server cleanly, with no outstanding transactions. If the application server does not shut down cleanly it performs *recovery* upon server restart and loads any required RAR files and native libraries. On completion of recovery, do not attempt any application-related work. Shut down the server and restart it. No further recovery is attempted by the application server on this restart, and normal application processing can proceed.

Install RAR

Use this page to install a resource archive (RAR) file in one of two ways. You can either upload a RAR file from the local file system, or specify an existing RAR file on a server. The RAR file must be installed at the node level, and you can select the node on this page.

To view this page in the administrative console click **Resources** > **Resource Adapters** > **Resource Adapters** > **Install RAR**.

For information about installing a resource adapter, see the topic, Installing a resource adapter archive (RAR) file.

Scope

Specifies the scope of the resource adapter. Only applications that are installed within this scope can use this adapter.

Local file system

Specifies the path of a RAR that resides on the same server as the console.

Data type String

Remote file system

Specifies the path of a RAR that resides on one of the nodes of the cell.

Data type String

Deploying SQLJ applications

Use Structured Query Language in Java (SQLJ) to develop data access applications that connect to DB2 databases. SQLJ is a set of programming extensions that enable you to use the Java programming language to embed statements that provide SQL (Structured Query Language) database requests.

About this task

The advantages of developing applications with SQLJ include improved performance and a shorter, more efficient development cycle. You can achieve the following with SQL:

- Improve performance by using static SQL statements.
- Reduce the development cycle:
  - Write less code with the simpler SQLJ syntax, which reduces the number of lines of code that is required to execute statements, set parameters, and retrieve parameters.

– Detect programming errors earlier in the development phase with the online check function, which performs data type validation and schema validation. See the DB2 documentation for a complete list of customization options.

Consider using SQLJ in situations where dynamic SQL is not needed, and where applications use DB2 as the database server.

The application server includes enhanced SQLJ support for applications that use container-managed persistence (CMP). The enhanced support includes the following items: include:

- Deploying CMP beans during the application installation in the application server.
- Customizing and binding SQLJ profiles with the administrative console or scripting.
- Customizing and binding SQLJ applications again without needing to reinstall the application.

These enhancements reduce the complexity of installing, deploying, and customizing SQLJ applications for both container-managed and bean-managed persistence.

Procedure

1. Acquire the required drivers to deploy an SQLJ application in the application server. You need the following files, depending on the JDBC provider that you use:

JDBC provider type	Required files
DB2 Using IBM JCC Driver This driver is also known as: • IBM Data Server Driver for JDBC and SQLJ • IBM DB2 Driver for JDBC and SQLJ • IBM DB2 Universal JDBC Driver.	db2jcc.jar or db2jcc4.jar
DB2 Universal JDBC driver (deprecated)	db2jcc.jar

2. Deploy the SQLJ application.
   - Deploy applications that use container-managed persistence (CMP):
     – “Deploying SQLJ applications that use container-managed persistence (CMP)” on page 184 with the DB2 Using IBM JCC Driver.
     – “Deploying SQLJ applications that use container-managed persistence (CMP) with the ejbdeploy tool” on page 185.
   - “Deploying SQLJ applications that use bean-managed persistence, servlets, or sessions beans” on page 186.
   - “Using embedded SQLJ with the DB2 for z/OS Legacy driver” on page 196 (deprecated).
3. Customize and bind the SQLJ profiles. Before the application server can use an SQLJ application, the SQLJ statements must be processed for the database server. By default, four DB2 packages are created in the database; one package is created for each isolation level. The customization process augments the profiles with information that is specific to the database. If you do not customize the SQLJ profiles, the SQLJ application uses dynamic SQL like a JDBC application.
   - “Customizing and binding profiles for Structured Query Language in Java (SQLJ) applications” on page 188.
   - Customize and bind SQLJ profiles with the wsadmin scripting tool. See the topic, Customizing and binding SQLJ profiles with the wsadmin tool.
   - “Customizing and binding SQLJ profiles with the db2sqljcustomize tool” on page 190.

Deploying SQLJ applications that use container-managed persistence (CMP)

Embed Structured Query Language in Java (SQLJ) statements in your applications to maximize the efficiency of transactions with your databases. Before your applications can take advantage of SQLJ, you must deploy the application and customize the SQLJ profiles that are created. The application server provides functionality to use SQLJ as the persistence mechanism for enterprise beans that use container-managed persistence. Deploy the CMP beans in the application server to enable SQLJ support.

Before you begin

You need an application that uses SQLJ and container-managed persistence. Develop this application in Rational Application Developer or another development tool.

About this task

Deploy SQLJ applications in the application server to simplify the process of SQLJ translation and bean deployment. The application server includes these new features for SQLJ support:

- Deploying CMP beans during the application installation in the application server.
- Customizing and binding SQLJ profiles with the administrative console or scripting.
- Customizing and binding SQLJ applications again without needing to reinstall the application.

You can also deploy the SQLJ application using the ejbdeploytool. Read the topic on deploying SQLJ applications that use container-managed persistence (CMP) with the ejbdeploy tool for more information.

Procedure

1. Create a top-down mapping to a DB2 database.
2. From your DB2 installation, copy the `sqlj.zip` file to a directory on your workstation.
3. Deploy the EAR file in the administrative console.
   a. Click **Applications** > **Install New application**.
   b. Select **Local file system** or **Remote file system**, and browse to the EAR file.
   c. Select **Detailed - Show all installation options and parameters**. Click **Next**.
   d. In **Step 1: Select installation options**, select **Deploy enterprise beans**. Configure any other options, and click **Next**.
   e. In **Step 3: Provide options to perform the EJB deploy**, select **SQLJ** for **Deploy EJB option - Database access type**.
   f. Enter the location of the `sqlj.zip` file in the **SQLj class path** field.
   g. Complete the installation process for the application.

What to do next

After the enterprise application is deployed, customize the SQLJ profiles using the administrative console, scripting, or the db2sqljcustomize tool:

- For administrative console support, read the topic on customizing and binding profiles for Structured Query Language in Java (SQLJ) applications.
- For scripting support, read the topic on the application management command group for the AdminTask object.
- For use of the db2sqljcustomize tool, read the topic on customizing and binding SQLJ profiles with the db2sqljcustomize tool.

Deploying SQLJ applications that use container-managed persistence (CMP) with the ejbdeploy tool

Embed Structured Query Language in Java (SQLJ) statements in your applications to maximize the efficiency of transactions with your databases. Before your applications can take advantage of SQLJ, you must deploy the application and customize the SQLJ profiles that are created. The application server provides functionality to use SQLJ as the persistence mechanism for enterprise beans that use container-managed persistence. Use the ejbdeploy tool to deploy the application.

About this task

You can deploy SQLJ applications with the ejbdeploy tool to deploy the enterprise application in a stand-alone environment.

Alternatively, the application server includes enhanced SQLJ support for applications that use container-managed persistence (CMP). The new features include:

- Deploying CMP beans during the application installation in the application server.
- Customizing and binding SQLJ profiles with the administrative console or scripting.
- Customizing and binding SQLJ applications again without needing to reinstall the application.

These enhancements reduce the complexity of installing, deploying, and customizing SQLJ applications for both container-managed and bean-managed persistence.Read the topic on deploying SQLJ applications that use container-managed persistence (CMP) for more information.

Procedure

1. Create a top-down mapping to a DB2 database.
2. From your DB2 installation, copy the `sqlj.zip` file to a directory on your workstation.
3. Modify the Java build path of your enterprise bean JAR project to include the `sqlj.zip` file.
4. Use Rational Application Developer or the DB2 SQLJ translator to automatically translate SQLJ.
   - Use Rational Application Developer:
     a. From the Project Navigator, click ***EJB_JAR_PROJECT_NAME*** > ***SOURCE_FOLDER*** > **META-INF** > **backends** > ***database_version***.
     b. Open Map.mapxmi in the Mapping editor.
     c. On the **Overview** panel, highlight the name of your JAR project in the Enterprise Beans column. You must highlight the name of the JAR project, not the name of one of the enterprise beans that is listed.
     d. On the **Properties** panel, expand **SQLJ**.
     e. Set **Is using SQLJ?** to `True`.
     f. Set **Translator Module** to the fully qualified path of the sqlj.zip file on your workstation.
     g. Save the Map.mapxmi file.
     h. Export the enterprise archive (EAR) file.
   - Use the DB2 SQLJ translator. This tool creates a .java version of your .sqlj file and a serialized profile, with a .ser extension, that is used later in processing. Refer to the DB2 documentation for more information on the SQLJ translator tool.
5. Deploy the EAR file with the ejbdeploy tool.
   a. Verify that the app_server_root/bin directory is in your class path.
   b. Run the ejbdeploy command utility with the `-sqlj` option. The ejbdeploy command will generate an EAR file with the name you specify and an Ant script with the name *application_name*.ear.xml.
      For example: :

```
ejbdeploy d:\application_name.ear
          working d:\deployed_application_name.ear
          -sqlj
          -dbvendor DB2UDB_V81
          -cp "C:\PROGRA~1\IBM\SQLLIB\java\sqlj.zip"
```

Note: Supply the location of the SQLJ translator sqlj.zip file with -cp, which is the class path option The ejbdeploy command does not access `sqlj.zip` from your system class path.

6. Choose the option for customization.
   - Use the application server's SQLJ support. Install the deployed application to customize the SQLJ profiles with the application server or scripting.
     a. Install the enterprise application in the application server.

        Note: Do not select **Deploy enterprise beans** during the application installation process in the administrative console. If you redeploy the enterprise beans from the administrative console, you will lose the customization changes that you have made.

     b. Customize the SQLJ profiles.
        – For administrative console support, read the topic on customizing and binding profiles for Structured Query Language in Java (SQLJ) applications.
        – For scripting support, read the topic on the application management command group for the AdminTask object.
   - Customize and bind the SQLJ profiles with the db2sqljcustomize tool. Read the topic on customizing and binding SQLJ profiles with the db2sqljcustomize tool.

Deploying SQLJ applications that use bean-managed persistence, servlets, or sessions beans

You can embed Structured Query Language in Java (SQLJ) statements in your applications to maximize the efficiency of transactions with your databases. Before your applications can take advantage of SQLJ, deploy the application and customize the created SQLJ profiles. You can use Rational Application Developer or the DB2 SQLJ translator to translate the application before deploying it on the application server.

Before you begin

Create an SQLJ application using Rational Application Developer or another development tool.

About this task

To deploy SQLJ applications that do not use container-managed persistence, translate the SQLJ application first to configure it for the application server environment. After translation, customize the SQLJ profiles in the application server, with scripting, or with the db2sqljcustomizer tool.

SQLJ support for applications that use bean-managed persistence include these features:

- Customizing and binding SQLJ profiles with the administrative console or scripting.
- Customizing and binding SQLJ applications again without reinstalling the application.

Procedure

1. Optional: Create a backup copy of your .java file. For example if your file is called `MyServlet.java`, copy `MyServlet.java` to `MyServlet.java.bkup`.
2. Optional: Rename your .java file to a file name with an `.sqlj` extension. For example, if your application is a servlet named `MyServlet.java`, rename `MyServlet.java` to `MyServlet.sqlj`

3. Optional: Edit the SQLJ file to convert the JDBC syntax to SQLJ syntax. When using SQLJ, if you want connection management for the application server to function properly, specify correct connection contexts.

   For example, convert the following JDBC operation:

   ```
   Connection con = dataSource.getConnection();
   Statement stmt = con.createStatement();
   stmt.execute("INSERT INTO users VALUES (1, 'user1')");
   con.commit();
   ```

   to the following SQLJ:

   ```
   // At the top of the file and just below the import statements, define Connection_Context
   #sql context Connection_context;
   .
   .
   Connection con = dataSource.getConnection();
   .
   .
   Connection_context ctx1 = new Connection_context(con);
   .
   .
   #sql [ctx1] {INSERT INTO users VALUES (1, 'user1')};
   .
   .
   con.commit(); ctx1.close();
   ```

   When you run the SQLJ translator, the `.java` file that is created has the same name as your old `.java` file. This provides you with a seamless transition to the SQLJ technology.
4. From your DB2 installation, copy the `sqlj.zip` file to a directory on your workstation. Modify the Java build path of your enterprise bean Java archive (JAR) file project to include the `sqlj.zip` file.
5. Use Rational Application Developer or the DB2 SQLJ translator to automatically translate SQLJ.
   - Use Rational Application Developer:
     a. In the Project Navigator, right-click your JAR project, and select **Add SQLJ Support...**.
     b. Select the check boxes for the applications for which you want SQLJ support.
     c. In the **SQLJ JAR file** field, type the fully qualified path to the `sqlj.zip` file that you previously copied to your workstation.
     d. Click **Finish**.
     e. Export the enterprise archive (EAR) file.
   - Use the DB2 SQLJ translator. This tool creates a `.java` version of the `.sqlj` file and a serialized profile, with an `.ser` extension, that is used later in processing. Refer to the DB2 documentation for more information about the SQLJ translator tool.
6. Package your JAR file for the enterprise application.
7. Install the application onto the application server, or customize the profiles with the db2sqljcustomize tool.
   - Customize the profiles with the application server.
     a. Package the JAR file for your enterprise beans, servlets, and any `.ser` files into an enterprise archive.
     b. Install the application in the application server, and customize SQLJ profiles with the administrative console or the wsadmin tool.

        Note: Do not select **Deploy enterprise beans** during the application installation process in the administrative console. If you redeploy the enterprise beans from the administrative console, you lose the customization changes that you have made.

        The application server provides enhanced support for SQLJ applications. Install the SQLJ application in the application server, and you can customize and bind SQLJ profiles through the administrative console or scripting:

  - To customize the SQLJ profiles with the administrative console, read the topic about customizing and binding profiles for Structured Query Language in Java (SQLJ) applications.
  - To customize SQLJ profiles with scripting, read the topic about the application management command group for the AdminTask object.
- To use the db2sqljcustomize tool, read the topic about customizing and binding SQLJ profiles with the db2sqljcustomize tool for more information.

Customizing and binding profiles for Structured Query Language in Java (SQLJ) applications

Simplify the process of customizing and binding SQLJ profiles for your applications by performing these functions in the administrative console or with scripting. SQLJ profiles must be customized and bound before the enterprise application can use the application's embedded SQL.

Before you begin

You must have an SQLJ application that has already been deployed and installed in the application server.

For SQLJ applications that use container-managed persistence, you can deploy the application in two ways:

- Deploy the SQLJ application in the application server. See the topic on deploying SQLJ applications that use container-managed persistence (CMP) for more information.
- Deploy SQLJ applications with the ejbdeploy tool. See the topic on deploying SQLJ applications that use container-managed persistence (CMP) with the ejbdeploy tool.

For SQLJ application that use bean-managed persistence, see the topic on deploying SQLJ applications that use bean-managed persistence, servlets, or session beans.

About this task

To take advantage of SQLJ applications in the application server, you need to customizing the SQLJ profiles that contain the embedded SQL statements. By default, four DB2 packages are created in the database; one for each isolation level. The customization process augments the profiles with information that is specific to the DB2 database. The database uses this information at run time.

In addition to profile customization, you need to bind the customized profiles to the DB2 database. Profile binding should only take place after the SQLJ profiles are customized.

You can also customize and bind profiles with scripting or the db2sqljcustomize tool:

- For scripting support, read the topic on the application management command group for the AdminTask object.
- For information on the db2sqljcustomize tool, read the topic on customizing and binding SQLJ profiles with the db2sqljcustomize tool for more information. If you customize profiles with the db2sqljcustomize tool, you will need to reinstall the application.

Procedure

1. Make sure the necessary database tables exist, as described in the topic on deploying data access applications.
2. Navigate to the SQLJ application that is installed in the application server. Click **Applications** > **Websphere enterprise applications** > ***app_name***.

   Note: Do not run multiple sessions of the administrative console to customize and bind profiles that are in the same EAR file.

3. Navigate to the SQLJ profiles section. Click **SQLj profiles**. When you click this link, the application server expands the EAR file for the application into a temporary directory; there might be a delay before the panel for SQLJ profiles is displayed.
4. Select **Customize and bind profiles** or **Bind packages**. Choose your option based on the profiles with which you are working:
   - If your profiles have not been customized, or you want to customize the profiles again, choose **Customize and bind profiles**.
   - If the profiles are already customized, choose **Bind packages**.
5. Choose to select profiles or a profile group to customize and bind.
   - Select profiles from the list that is provided.
     a. Select the profiles from the list and click **Add**. The list displays the SQLJ profiles that are present in the enterprise application.

        Note:
        – Select more than one profile by holding CTRL.
        – Select a contiguous list of profiles by selecting the first profile name, holding SHIFT, and selecting the last profile. You will select the first profile, last profile, and any profiles in the middle.

     b. Select **Customize/bind the selected SQLj profiles as a group** This option specifies that the application server will create a .grp file that contains the SQLj profiles that are processed. You can use the .grp file for other binding operations in the future. After you have completed this panel and click **OK**, you will be given an option to download the .grp file.
   - Select **Use a profile group file to specify profiles to customize/bind**. Select this to specify a profile group to process. Click **Browse...** to locate the file on the system.
6. Complete the necessary information to connect to the database. You need to complete the following fields:

   Database URL
   Specifies the URL of the database to which the profile/s will be bound. The typical syntax is:
   `jdbc:db2://<host name="">:<port>/<database name="">.</database></port></host> or`

   or
   `fully_qualified_host_name``:`*`port`*

   User Specifies the user ID for the database administrator on the server where the database is located.

   Password
   Specifies the password for the database administrator on the server where the database is located.

   Additional options
   Specifies additional options to use during the customization and bind processes. See the DB2 documentation for a complete list of customization options.

   Class path
   Specifies the class path where sqlj.zip, and db2jcc.jar or db2jcc4.jar are located.
7. Click **OK**.

   Note: If you are processing a large enterprise application, or you are processing many SQLJ profiles, the process might take longer than the default timeout for the administrative console. The default connection timeout for the application server's administrative console is set to 30 minutes. If the default timeout is reached and you lose the connection to the server, you can check the system output log for the final results of the customization and bind process.

To prevent this disconnection, configure the console session timeout to a longer period of time. After a successful customization and binding process, check the system output log for the total processing time. Use that time period as a basis for the new timeout value. For information about how to configure the console timeout, see the topic on changing the console session expiration.

Results

After the application server finishes processing the SQLJ profiles, you will see the results from the customization and binding. The results panel displays messages from the database server, as well as summary results from the application server.

If the operation completed successfully, the following message will be printed to the system log:

```
ADMA0507I=ADMA0507I: The SQLJ operation on application {0} completed successfully.  Exit code: {1}
ADMA0507I.explanation=This informational message indicates the program status.
ADMA0507I.useraction=No user action is required.
```

If the operation did not complete successfully, the following message will be printed to the system out log:

```
ADMA0506I=ADMA0506I: The SQLJ operation on application {0} did not complete successfully.  Exit code: {1}
ADMA0506I.explanation=The SQLJ operation encountered a problem.  This informational message indicates
the program status. Prior messages in the command output give details of the problem.
ADMA0506I.useraction=Check the command output for the cause of the problem.
```

Customizing and binding SQLJ profiles with the db2sqljcustomize tool

Customize and bind SQLJ profiles with the db2sqljcustomize tool before you install the SQLJ application in the application server.

Before you begin

To perform this task, you must have SQLJ application that has been deployed, but the application should not be installed in the application server. If the application is already installed in the application server, you will need to reinstall the application after you customize the profiles. You also need serialized profiles for the SQLJ application.

For SQLJ applications that use container-managed persistence, you can deploy the application in two ways:

- Deploy the SQLJ application in the application server. See the topic on deploying SQLJ applications that use container-managed persistence (CMP) for more information.
- Deploy SQLJ applications with the ejbdeploy tool. See the topic on deploying SQLJ applications that use container-managed persistence (CMP) with the ejbdeploy tool.

For SQLJ application that use bean-managed persistence, see the topic on deploying SQLJ applications that use bean-managed persistence, servlets, or sessions beans.

About this task

To take advantage of SQLJ applications in the application server, you need to customize the SQLJ profiles. The customization process augments the profiles with information that is specific to the DB2 database. The database uses this information at run time. By default, four DB2 packages are created in the database; one package is created for each isolation level.

The application server supports customizing and binding the SQLJ profiles in the administrative console or with scripting:

- For administrative console support, read the topic on customizing and binding profiles for Structured Query Language in Java (SQLJ) applications.

- For scripting support, see the topic on the application management command group for the AdminTask object.

Procedure

1. Make sure the necessary database tables exist, as described in the topic on deploying data access applications.
2. Transfer the serialized profiles to the environment on which you installed your application. Alternatively, use the Java `jar` command to extract the serialized profiles from the JAR file in your installed EAR directory.
3. Add the location for the SQLJ profiles and the application's JAR file to your environment's class path.
4. Make sure the necessary database tables exist, as described in the topic on deploying data access applications.
5. Optional: If your application is not running in a clustered environment, you can use the Ant script to make customization easier. If you run a batch SQLJ customization against an EAR file with the ejbdeploy tool, the tool produces an Ant script that is named *application_name*.ear.xml. You can use this script file to run the DB2 customizer program against the serialized profiles in all of the enterprise bean JAR files for the associated EAR file. The script updates each enterprise bean's JAR file with a serialized profile and replaces the JAR files in the existing EAR file with the modified versions.
   a. Change the values of the database URL, and the database user and password properties in ejbdeploy.sqlj.properties. This file is a common file to all Ant scripts that are generated by the ejbdeploy command. The ejbdeploy.sqlj.properties script defines the global properties for:
      - Database URL - db.url
      - User - db.user
      - Password - db.password

      The Ant script uses the URL, user, and password properties in the serialized profile to customize the profile. By default, the properties for the serialized profile are created from the global properties.
   b. Run the Ant script, specifying the `properties` target. For example:

      ```
      ws_ant -buildfile application_name.ear.xml properties
      ```

      This script creates the properties file, *application_name*.ear.properties. The *application_name*.ear.properties file contains properties that specify the default names for the packages corresponding to each serialized profile in the EAR file. This is a sample properties file:

      ```
      url.MyEJB1.jar.DB2UDBNT_V8_1=jdbc:db2://localhost:50000/MyDB1
      user.MyEJB1.jar.DB2UDBNT_V8_1=dbuser
      password.MyEJB1.jar.DB2UDBNT_V8_1=dbpassword
      pkg.MyEJB1.jar.DB2UDBNT_V8_1=TEST
      url.MyEJB2.jar.DB2UDBNT_V8_1=jdbc:db2://localhost:50000/MyDB2
      user.MyEJB2.jar.DB2UDBNT_V8_1=dbuser
      password.MyEJB2.jar.DB2UDBNT_V8_1=dbpassword
      pkg.MyEJB2.jar.DB2UDBNT_V8_1=WORK
      ```
   c. Use the DB2 Control Center to identify the packages that are installed in the database. The DB2 SQLJ customizer requires a type 4 database URL in the form of:

      ```
      jdbc:db2://host-name:port/database-name
      ```

      It also requires a user and password. The value of the port is 50000, unless you change it when you install DB2.
   d. Change the names that are used by the script file to ensure that the names for each customization profile do not conflict with existing package names that are in the database. Ant scripts that are generated for different EAR files use the same package names by default, and the script will overwrite existing packages unless you change the names. Overwritten packages can cause errors at run time.

DB2 uses the first seven characters of the package name. The DB2 customizer uses this name to create four packages in the database. For example, if you specify the name TEST, the DB2 customizer will create packages called TEST1, TEST2, TEST3, and TEST4.

e. Run the Ant script. The Ant script updates the original EAR file with the modified serialized profiles.

Note: Verify that you have db2jcc.jar in the class path. This file should have been added to the class path environment variable when DB2 V8 FixPak1 was installed.

A sample Ant command looks like this:

```
ws_ant -Dwork.dir=tmp
       -Dscript.property.file=other.properties
       -buildfile application_name.ear.xml
```

where:

- `-buildfile` specifies the XML file to create.
- `-Dscript.property.file` specifies a different properties file. This parameter is optional. If you want your Ant script to use a another file instead of *application_name*.ear.properties, specify the Dscript.property.file property when you run the script.
- `-Dwork.dir` specifies a temporary working directory for the script. The script will create and delete files and subdirectories in this directory. If the working directory contains existing files and directories with the same name as the files and directories used by the script, the script will erase or overwrite the files and directories. This script creates and uses a directory called `tmp` as its working directory.

f. Proceed to installing the application in the application server..

6. Run the db2sqljcustomize tool to customize the SQLJ profiles that correspond to each enterprise bean's JAR file. When you generate your deployment code, serialized profiles (files with a .ser extension) that are specific to your application are created. These profiles exist in the same directory as your SQLJ files, and the files must be customized to the environment before they can be used. When you run the DB2 SQLJ customizer against the serialized profiles, you create static SQL in the database that DB2 will use at run time. The customization phase creates four database packages that contain static SQL, one for each isolation level.

a. Optional: Consider using the SQLJ customizer tool to enable context caching for your application's data source connections. DB2 V8.1 fix pack 6 provides the new caching option with the db2sqljcustomize tool called db2optimize. You can run this option if your application uses the explicit connection context instead of the default context.

Note:

- SQLJ context caching support requires the DB2 with IBM JCC driver or Version 2.2 or later of the DB2 Universal JDBC Driver with APAR PQ87786 applied.
- If you want to enable context caching for an application or BMP bean that caches connections across transaction boundaries, you cannot use shareable connections. Use the get/use/close pattern of connection usage when you invoke the db2optimize option, or an object closed exception occurs. The following code gives an example of incorrect connection usage for context caching:

```
utx.begin();
     cons =ds.getConnection(
     request.getParameter("db.user"),
     request.getParameter("db.password"));
         cmctx1 = new CM_context(cons);
         #sql [cmctx1] {DELETE FROM cmtest WHERE id=1};
 utx.commit();
       //The next statement verifies the result:
         #sql [cmctx1] cursor1 = {SELECT id, name FROM cmtest WHERE id=1};
```

In this case, the `Select` statement elicits an object closed exception. To prevent the exception from occurring, close the connection before committing the transaction. Then get a new connection and a new context before running the `Select` statement.

The following example code demonstrates proper syntax for running the option on the serialized profile:

```
sqlj -db2optimize SQLJTransactionTest.sqlj
db2sqljcustomize -url jdbc:db2://localhost:50000/dbname -user USER_NAME -password PASSWORD
SQLJTransactionTest_SJProfile0.ser
```

b. Run the db2sqljcustomize tool to customize the SQLJ profiles. After you successfully run the db2sqljcustomize command, customized profiles exist in the directory from which you issued the command. If you run the db2sqljcustomize command from the directory that contains the serialized profiles that were not customized, the customized versions will overwrite previous versions that have the same file names.

   The recommended syntax for running the db2sqljcustomize command is:

   ```
   db2sqljcustomize -url JDBC_URL -user USER_NAME  -password PASSWORD
   [-rootpkgname PACKAGE_NAME] SERIALIZED_PROFILE1 SERIALIZED_PROFILE2 ...
   ```

   where:

   - *JDBC_URL* is the JDBC URL that is used to access the DB2 system where your tables reside.
   - *USER_NAME* is a valid user name for the DB2 system where your tables reside.
   - *PASSWORD* is the password for the specified user name.
   - *PACKAGE_NAME* is a valid partitioned data set (PDS) member name, up to seven characters long. Each of the four packages that are created by the profile customizer begin with this name and are appended with a number from 1 to 4. If you customize only one serialized profile, this value defaults to a shortened version of the serialized profile name and the -rootpkgname parameter is not required. If you customize more than one serialized profile with the same command, there is no default value and the -rootpkgname parameter is required.
   - *SERIALIZED_PROFILE#* is the name of the serialized profile that you are customizing.
     – To customize more than one serialized profile with the same command, list multiple files, separated by spaces.
     – Alternatively, you can specify the `-rootpkgname` parameter to customize more than one serialized profile with the same command.

   Note: The following options provide more control over the customization process:

   - `-automaticbind yes` specifies to run the DB2 SQLJ customizer against the serialized profiles to create static SQL in the database that the database will use at run time. The customization phase creates four database packages that contain static SQL, one for each isolation level.
   - `-onlinecheck NO and -bindoptions "VALIDATE RUN"` specifies settings to bypass errors during a profile customization and ensure a successful customization.

7. Update the JAR file for the enterprise beans with the serialized profiles.
8. Use the `jar` command to replace the serialized profiles in your JAR file with the customized profiles.

   Note: The customized files must be placed in a location that is part of the application class path, and they must exist ahead of the serialized profiles that are not customized in your JAR file. If you decide to replace the serialized profiles in your JAR file, maintain the directory structure in which the profiles exist.
9. Package the JAR file for the enterprise bean, servlets, and serialized profiles into an enterprise archive (EAR) file.
10. Install the application in the application server.

Note: Do not select **Deploy enterprise beans** during the application installation process in the administrative console. If you redeploy the enterprise beans from the administrative console, you will lose the customization changes that you have made.

SQLJ profiles and pureQuery bind files settings

Use this panel to do customization and binding for the Structured Query Language in Java (SQLJ) profiles for DB2 that are included in this application. You can also use this panel to do binding for pureQuery bind files in the application. You can view SQLJ profiles for other database types, but you cannot change these profiles. PureQuery bind files are only valid for DB2. Use SQLJ or pureQuery to develop data access applications that connect to DB2 databases. SQLJ is a set of programming extensions that enable a programmer to use the Java programming language to embed statements that provide SQL database requests. PureQuery provides an alternate set of APIs that can be used instead of JDBC to access the DB2 database.

To view this administrative console page, click **Applications** > **Websphere enterprise applications** > ***app_name*** > **SQLJ profiles and pureQuery bind files**.

Advantages of developing applications with SQLJ include improved performance and a shorter, more efficient development cycle. With SQLJ, you can:

- Improve performance by using static SQL statements.
- Reduce the development cycle by:
  - Writing less code with the simpler SQLJ syntax, which reduces the amount of code that is required to execute statements, and set and retrieve parameters.
  - Detecting programming errors earlier in the development phase with the online check function, which performs data type and schema validation. Activate this function by running it as an option with the **db2sqljcustomize** command. See the DB2 documentation for a complete description of the SQLJ customize command.

DB2 pureQuery run time is an alternative set of APIs to JDBC or SQLJ. Advantages of developing applications with pureQuery include allowing SQL execution to be either dynamic or static. In addition to improved performance by using static SQL statements, pureQuery has better problem determination and diagnosis because it allows for errors at the DB2 server to be related back to application artifacts rather than to SQL that was generated by an application generator.

Customize and bind profiles:

Specifies that the application server processes the SQLJ profiles that you select from this application.

Note: This selection does not apply to pureQuery. If selected, this option is ignored when processing pureQuery bind files.

By default, one DB2 package is created in the database for each isolation level. The customization process augments the profile or profiles with information that is specific for the DB2 database for use at run time. Typically, the customization process should run after the SQLJ application has been translated and before the application is started. If you do not run the customization step, the SQLJ application uses dynamic SQL like a JDBC application.

Binding DB2 SQLJ profiles involves the process of binding the customized SQLJ profiles to the DB2 database.

Bind packages:

Specifies that the application server binds the SQLJ profiles that you select to the DB2 database server.

Note: This selection does not apply to pureQuery. If selected, this option is ignored when processing pureQuery bind files.

Bind packages from the SQLJ application that have already been customized.

Select and order the profiles to customize/bind:

Specifies the profiles to process from the list that is provided.

- Select a profile or group of profiles from the **Available profiles**, and click **Add** to add the profile that is selected to **Selected Profiles**.
- Select a profile or group of profiles from the **Selected Profiles**, and click **Remove** to add the profile that is selected to **Available profiles**.

When SQLJ or pureQuery profiles have been added to **Selected Profiles**, select profiles from that list and use **Move Up** or **Move Down** to change the order in which the profiles are processed.

Customize/bind the selected SQLJ profiles as a group:

Specifies that the application server creates a .grp file that contains the SQLJ profiles that you selected.

Note: This selection does not apply to pureQuery. If selected, this option is ignored when processing pureQuery bind files.

When you click **OK**, there is an option on the next panel to download the .grp file.

Use a profile group file to specify profiles to customize/bind:

Specifies a profile group file from the local file system to customize or bind.

Database URL:

Specifies the URL of the database to which the profile or profiles are bound.

The typical syntax is:

```
jdbc:db2://host_name:port_name/database_name
```

User:

Specifies the user ID for the database administrator on the server where the database is located.

Password:

Specifies the password for the database administrator on the server where the database is located.

Additional options:

Specifies additional options to use during the customization and bind processes.

Options for pureQuery binding uses the following syntax:

```
-bindoptions "BLOCKING NO"
```

For more information about pureQuery bind options, refer to the DB2 pureQuery Bind Utility topic.

Class path:

Specifies the class path where the sqlj.zip, and db2jcc.jar or db2jcc4.jar files for SQLJ are located. Specifies the class path where the pdq.jar, pdqmgt.jar, db2jcc.jar, db2jcc_license_cisuz.jar files for pureQuery are located.

Download SQLJ profile group

Use this panel to download the group file for the Structured Query Language in Java (SQLJ) profiles that are bound together as a single package on the DB2 database server. You can use the file when performing future customization or binding work on the application. Click the link that is provided to download the profile group to your local file system. The group file has a filename extension of .grp and a HTTP Content-Type of text/plain.. Your web browser settings might cause the browser to display the file contents rather than prompting you for a download destination. If this happens, you can manually copy and paste the contents into your own .grp file.

Note: This topic does not apply to IBM Optim™ PureQuery Runtime. IBM Optim PureQuery Runtime does not support binding pureQuery bind files as a group.

Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***app_name*** > **SQLJ profiles and pureQuery bind files**. When you are selecting the profiles to customize and bind, select **Customize/bind the selected SQLJ profiles as a group** to view this console panel.

Review results

Use this panel to review the results from the customization and binding process for the Structured Query Language in Java (SQLJ) profiles or pureQuery bind files. Use SQLJ or IBM Optim PureQuery Runtime to develop data access applications that connect to DB2 databases. SQLJ is a set of programming extensions that enable a programmer to use the Java programming language to embed statements that provide SQL (Structured Query Language) database requests. IBM Optim PureQuery Runtime provides an alternate set of APIs that can be used instead of JDBC to access the DB2 database.

Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **SQLj profiles and pureQuery bind**. Select profiles to customize and bind, complete the necessary fields, and click **OK** to view this console panel.

Review results:

Displays the results of the customization and bind process. The field shows information that is received from the database and summary statements from the application server.

Using embedded SQLJ with the DB2 for z/OS Legacy driver

Structured Query Language in Java (SQLJ) is a set of programming extensions that enable a programmer, using the Java programming language, to embed statements that provide Structured Query Language (SQL) database requests. You can use the DB2 for z/OS Legacy driver with your data access applications.

About this task

Notes:

1. To use SQLJ with WebSphere Application Server for z/OS and the DB2 for z/OS Legacy Driver, install DB2 APAR PQ76442.
2. Container Managed Persistence (CMP) beans generated using SQLJ are not supported by the DB2 for z/OS Legacy Driver. Use the DB2 Universal Driver for CMPs that are generated using SQLJ.

Following are the steps required to develop applications with SQLJ that run on WebSphere Application Server for z/OS v6.0 using the DB2 for z/OS Legacy driver.

Procedure

1. Design your application in Rational Application Developer according to your requirements, using SQLJ when necessary. For example, if you develop a bean called Test that uses BMP, code TestBean.sqlj (instead of TestBean.java).

a. From your DB2 for z/OS installation, copy the `db2sqljclasses.zip` file to a directory on your workstation, then modify the Java Build Path of your EJB Java archive (JAR) project to include the `db2sqljclasses.zip` file.

b. Translate your SQLJ code according to the following steps:

   1) Locate your SQLJ file, then use ASCII mode transfer to FTP it to an HFS in your z/OS environment.

   2) Use the sqlj command to translate your SQLJ code into Java code. Two files are produced, one with a `.java` extension and the other with an `.ser` extension.

      ```
      sqlj -compile=false SQLJ_FILE_NAME
      ```

   3) Use ASCII mode transfer for the `.java` file and BINARY mode transfer for the `.ser` file to move these files back to the directory on your workstation where the SQLJ file resides.

   4) Refresh the project.

c. Generate deployment code for your application.

d. Export your EAR file.

2. Install your application

   a. Create a data source with the DB2 for zOS Local JDBC Provider (RRS). When you define your JDBC Provider and data source, the default values are sufficient for providing SQLJ support.

   b. Install your application into WebSphere Application Server.

      Use the data source you created in Step 1 to resolve your resource references.

3. Customize your serialized profiles When you generate your deployment code, serialized profiles, or files with an `.ser` extension, that are specific to your application, are created. These profiles must be customized in a z/OS environment before they can be used.

   a. Use binary transfer to transfer the serialized profiles to the z/OS environment on which you installed your application. Alternatively, use the Java jar command to extract the serialized profiles from the EJB JAR file in your installed EAR directory.

   b. Use the db2profc command to customize your serialized profiles. You can get information about the various options associated with this command from the DB2 documentation; however, here are the minimum requirements to customize your profile:

      ```
      db2profc -pgmname=PROGRAM_NAME PROFILE_NAME
      ```

      - Where:
        - *PROGRAM_NAME* must be a valid MVS™ PDS member name, and can be up to seven characters.
        - *PROFILE_NAME* is the name of the serialized profile that you want to customize. You must run db2profc one time for each profile.
      - The profile customizer creates four DBRM data sets in the PDS *USERNAME*.DBRMLIB.DATA. The member names of the DBRMs begin with what you specified as *PROGRAM_NAME*.
      - Ensure that your CLASSPATH environment variable includes:
        - The location of the serialized profile
        - The EJB JAR file in your installed EAR directory
      - Allocate a PDS to contain the DBRMs that are created. Name this PDS *USERNAME*.DBRMLIB.DATA, where *USERNAME* is the user who implements the db2profc command.

        The following fields are an example:

        ```
        Space units=TRACK
        Primary quantity=15
        Secondary quantity=5
        Directory blocks=10
        Record format=FB
        Record length=80
        Block size=27920
        Data set name type=PDS
        ```

c. Place the existing serialized profiles, which are now customized, into a location that is part of the application classpath and that is ahead of the serialized profiles that exist in your EJB JAR file.

   The output of the DB2 profile customizer and the input file have the same name. Move the output file ahead of the original serialized profile in the classpath. Alternatively, you can move the customized profile into the EJB JAR file, replacing the original. It is recommended that you replace the original file.

   IMPORTANT: If you run the db2profc command from the directory where the serialized profile exists, the profile customizer overwrites the serialized profile. Because you need only the customized version after the profile customizer has run, this is not a problem.

d. Bind your DBRMs into a package.

   Note: You must create your database tables before binding your DBRMs. If you do not, the bind job fails.

   The db2profc customization command creates a series of DBRMs that must be bound into packages. For each customized profile, four DBRMs are created.

   These DBRMs:

   - Are located in *USERNAME*.DBRMLIB.DATA
   - All have names that begin with what you specified as *PROGRAM_NAME*
   - Are numbered from 1-through-4

     For example, if you log in as IBMUSER, and you specify -pgmname=TESTBMP, then run the db2profc command, the four data sets, TESTBMP1, TESTBMP2, TESTBMP3, AND TESTBMP4 are created and placed in the PDS IBMUSER.DBRMLIB.DATA.

     These data sets must be bound into packages with isolation of UR, CS, RS, and RR. You must run a bind for each serialized profile that you customize.

e. After you bind all of the DBRMs into packages, bind the packages into a plan. Name the plan whatever you like.

   IMPORTANT: You must also include the JDBC packages in the package list (PKLIST) of your new plan. The default names for the JDBC packages to include are DSNJDBC.DSNJDBC1, ..., DSNJDBC.DSNJDBC4. If your installation did not use the default names for the JDBC packages, contact your DB2 administrator to determine the names of the JDBC packages that you need to include.

   Following is a sample job used to bind a new plan.

   - One serialized profile was created while logged on as IBMUSER.
   - **-pgmname=TESTBMP** was specified to run db2profc.
   - The new plan is named SQLJPLAN.

```
//BBOOLS  JOB (516B,1025),'IBMUSER',MSGCLASS=H,CLASS=A,PRTY=14,
//         NOTIFY=&SYSUID,TIME=1440,USER=IBMUSER,PASSWORD=IBMUSER,
//         MSGLEVEL=(1,1)
//*********************************************************************
//BINDOLS  EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB  DD  DSN=IBMUSER.DBRMLIB.DATA,DISP=SHR
//*        DD  DSN=MVSDSOM.DB2710.SDSNDBRM,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *

 DSN SYSTEM(DB2)

 BIND -
     PACKAGE(TESTBMP) -
     QUALIFIER(IBMUSER) -
     MEMBER(TESTBMP1) -
     VALIDATE(BIND) -
     ISOLATION(UR) -
```

```
    SQLERROR(NOPACKAGE) -

BIND -
    PACKAGE(TESTBMP) -
    QUALIFIER(IBMUSER) -
    MEMBER(TESTBMP2) -
    VALIDATE(BIND) -
    ISOLATION(CS) -
    SQLERROR(NOPACKAGE) -

BIND -
    PACKAGE(TESTBMP) -
    QUALIFIER(IBMUSER) -
    MEMBER(TESTBMP3) -
    VALIDATE(BIND) -
    ISOLATION(RS) -
    SQLERROR(NOPACKAGE) -

BIND -
    PACKAGE(TESTBMP) -
    QUALIFIER(IBMUSER) -
    MEMBER(TESTBMP4) -
    VALIDATE(BIND) -
    ISOLATION(RR) -
    SQLERROR(NOPACKAGE) -


BIND PLAN(SQLJPLAN)                -
 QUALIFIER(IBMUSER)                -
 PKLIST(TESTBMP.*                  -
        DSNJDBC.*                ) -
 ACTION(REPLACE) RETAIN            -
 VALIDATE(BIND)

END
/*
```

f. Grant the appropriate authority to your new plan. Use an interface to DB2, such as SPUFI, to grant the authority. Issue this command:

```
GRANT EXECUTE ON PLAN PLANNAME TO APPSERVERID
```

Where:

- *PLANNAME* is the name of the plan that you bound.
- *APPSERVERID* is the ID under which WebSphere Application Server runs; for example, CBSYMSR1.

4. Configure your data source to use your new plan
   a. From the WebSphere Application Server for z/OS Administrative Console, navigate to your Data Source and select Custom Properties.
   b. Select the Custom Property **planName**.
   c. Update the value of **planName** with what you named your plan when it was bound.
   d. Set **enableSQLJ** to **true**.
5. Stop and restart your server.
6. Run your application.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 16. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*`x`* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/`*`profile_name`* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express` directory.

The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Administering data access applications

These administrative tasks consist primarily of configuring the objects, or resources, through which applications connect with a backend, and tuning those resources to handle the volume of connection requests.

Procedure

1. If your application contains web modules or EJB modules that require access to a backend, configure resources according to your type of enterprise information system (EIS):
   - For a relational database, follow the steps outlined in the topic, Configuring a JDBC provider and data source. If you are using a DB2 database, the topic, Configuring an application to use pureQuery is another option. IBM Optim PureQuery Runtime provides an alternative to JDBC as a way to access the DB2 database.
   - For a non-relational database, or another type of EIS such as the Customer Information Control System (CICS), you must configure a resource adapter and connection factories. The topic, Accessing data using Java EE Connector Architecture connectors, provides information on setting up these objects.

   Note: When you specify the Java Naming and Directory Interface (JNDI) name for resources, adhere to the following requirements:
   - Do not assign duplicate JNDI names across different resource types (such as data sources versus J2C connection factories or JMS connection factories).
   - Do not assign duplicate JNDI names for multiple resources of the same type in the same scope.
2. Configure an authentication alias for the new web module resource or EJB module resource only if the application code, rather than WebSphere Application Server, authenticates connections with the backend. This security configuration is called component-managed authorization, and is indicated in the application deployment descriptor as `res-auth = Application`.

   Container-managed authorization, which is designated as `res-auth = Container`, indicates that Application Server performs signon for backend connections. The container-managed authentication alias must be specified on the application resource reference. This task can be done during application assembly or deployment, along with mapping the resource reference to a data source or connection factory resource. After application deployment, however, you can alter the container-managed authentication alias using the administrative console. Click **Applications** > **Websphere enterprise applications** > ***application_name***, and select the link to the appropriate mapping page. For example, if you want to alter the alias of an EJB module resource, you might click **Map data sources for all 2.x CMP beans**. For a web module resource, click **Resource References**.

   Read the J2EE connector security topic for detailed reference on resource authentication.
3. If your application contains a client module that requires data access, see the topic, Configuring data access for application clients. In this single configuration process, you can define authentication data for either component-managed or container-managed signon.
4. Specify connection pool settings.

5. Test a connection to the new data source. See the article, Test connection service, for information on the available methods for testing connections. This article also addresses important data source settings that can affect the accuracy of your test connection results.
6. Set the JDBC trace service. The JDBC trace log information augments the JVM log data for data source failures.

   To activate the trace using the administrative console, read the topic, Enabling trace at server startup.. Specify **WAS.database** as the trace group and select **com.ibm.ws.db2.logwriter** as the trace string.
7. Gather connection pool statistics by activating the JDBC connection pool counters or the J2C connection pool counters. Alternatively, you can use Performance Monitoring Infrastructure (PMI) method calls to gather connection statistics; see the topic, Connection and connection pool statistics.
8. Tune your database to accommodate the connection volume. If you use DB2 UDB for iSeries®, see the topic, DB2 Universal Database performance tips, as a starting point reference.

Configuring Java EE Connector connection factories in the administrative console

To access an enterprise information system (EIS), configure connection factories, which instantiate resource adapter classes for establishing and maintaining resource connections.

About this task

An application component uses a connection factory to access a connection instance, which the component then uses to connect to the underlying enterprise information system (EIS). Examples of connections include database connections, Java Message Service connections, and SAP R/3 connections.

Procedure

1. Click **Resources** > **Resource Adapters** > **Resource adapters**.
2. In the **Resource adapters** panel, select the resource adapter that you want to configure.
3. From the **Additional Properties** heading, click **J2C connection factories**.
   a. Click **New**.
   b. Specify any properties for the connection factory in the **General Properties** panel.
   c. Select the authentication preference.
   d. Select the aliases for **Component-managed authentication**, **Container-managed authentication**, or both. Some choices for the mapping-configuration alias do not use a container-managed authentication alias, so you will not be able to select a container-managed alias if one of those mapping-configuration aliases is selected.

      If you have defined security domains in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. Security domains allow you to isolate authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that will be able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them.

      Note: If the resource adapter supports XA, an option for **Authentication alias for XA recovery** will be available.

      If there are no aliases that are available, or you want to define a different alias:
      1) Click **Apply** to save the current settings.
      2) Click **JAAS - J2C authentication data** from the **Related Items** heading.
      3) Click **New**.
      4) Define the properties for the alias in **General Properties**.
      5) Click **OK**.
   e. Click **OK**.

4. Click the name of the J2C connection factory that you created.
5. From the **Additional Properties** heading, click **Connection pool properties**.
   a. Change any values by clicking the property name. For more information on the settings for connection pools, read the topic Tuning connection pools, or the topic, Connection pool settings.
   a. Click **OK**.
6. Click **Custom properties** from the **Additional Properties** heading.
   a. Click any property name to change its value. If the **UserName** and **Password** properties are defined, they will be overridden by the component-managed authentication alias that you specified in the previous step.
   b. Click **Save**.
7. Restart the application server for the changes to take effect.

Configuring connection factories for resource adapters within applications

To access an enterprise information system (EIS), configure connection factories, which instantiate resource adapter classes for establishing and maintaining resource connections.

About this task

An application component uses a *connection factory* to access a connection instance, which the component then uses to connect to the underlying enterprise information system (EIS). Examples of connections include database connections, Java Message Service connections, and SAP R/3 connections.

Procedure

1. Optional: Install the application if it is not already installed on the application server.
   a. Click **Applications** > **Install New Application**.
   b. Browse to find the appropriate EAR file, which contains an RAR file.
   c. Click **Next**.
   d. Select **Resource ref mapping to a J2C Connection Factory**, then click **Next**.
   e. In **Step 2: Map module to servers**, select the resource adapters with which to associate your application and click **Next**.
   f. Complete the installation process for the application. For more information on installing applications, refer to the topic, Installing enterprise application files with the console.
2. Select the application that you want to configure.
3. Click **Modules** > **Manage Modules**.
4. Select the name of the RAR file in the **Manage Modules** panel.
5. Click **Resource Adapter** under the **Additional Properties** heading.
6. Under the **Additional Properties** heading, click **J2C connection factories**.
   a. Click **New**.
   b. Specify any properties for the connection factory in the **General Properties** panel.
   c. Select the authentication preference.
   d. Select an alias for **Component-managed authentication** if any application components with *Application* or *Per connection factory* authentication specified in the resource reference are going to be getting connections from this connection factory using the empty-argument getConnection() method. For resources that support XA, you can specify an Authentication alias for XA recovery. If there are no aliases that are available, or you want to define a different alias:
      1) Click **Apply** to save the current settings.
      2) Click **JAAS - J2C authentication data** under the **Related Items** heading.
      3) Click **New**.
      4) Define the properties for the alias in **General Properties**.

5) Click **OK**.

e. Click **OK**.

7. Click the name of the J2C connection factory that you created.
8. Under the **Additional Properties** heading, click **Connection pool properties**.
   a. Change any values by clicking on the property name. For more information on the settings for connection pools, refer to the topic, Tuning connection pools, or the topic, Connection pool settings.
   a. Click **OK**.
9. Click **Custom properties** under the **Additional Properties** heading.
   a. Click any property name to change its value. If the **UserName** and **Password** properties are defined, they will be overridden by a **component-managed authentication** alias that you might have configured.
   b. Click **Save**.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 17. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, was.install.library, and is located under the *app_server_root* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the /QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/*profile_name* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server - Express is the /QIBM/UserData/WebSphere/AppServer/V8/Express directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Connection pool settings

Use this page to configure connection pool settings.

This administrative console page is common to JDBC data sources and JMS connection factories (unified, queue, or topic connection factories). To view this page, the path depends on the type of resource, but generally you select an instance of the resource type then click **Connection Pool**. For example:

- Click **Resources** > **JDBC** > **Data Sources** > ***data_source*** > **[Additional Properties] Connection pool properties**
- Click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Connection pool**

Note: Connection pooling is not supported in an application client. The application client calls the database directly and does not go through a data source. If you want to use the getConnection() request from the application client, configure the JDBC provider in the application client deployment descriptors, using Rational Application Developer or an assembly tool. The connection is established between application client and the database. Application clients do not have a connection pool, but you can configure JDBC provider settings in the client deployment descriptors.

Connection timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

This value indicates the number of seconds that a connection request waits when there are no connections available in the free pool and no new connections can be created. This usually occurs because the maximum value of connections in the particular connection pool has been reached.

For example, if Connection timeout is set to 300, and the maximum number of connections are all in use, the pool manager waits for 300 seconds for a physical connection to become available. If a physical connection is not available within this time, the pool manager initiates a `ConnectionWaitTimeout` exception. In most cases, you should not retry the getConnection() method; if a longer wait time is required you should increase the Connection timeout setting value. If a `ConnectionWaitTimeout` exception is caught by the application, review the expected connection pool usage of the application and tune the connection pool and database accordingly.

If the Connection timeout is set to 0, the pool manager waits as long as necessary until a connection becomes available. This happens when the application completes a transaction and returns a connection to the pool, or when the number of connections falls below the value of Maximum Connections, and a new physical connection is created.

If Maximum Connections is set to 0, an infinite number of physical connections are enabled, and the Connection timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Maximum connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. When this number is reached, no new physical connections are created. The requester waits until a physical connection that is currently in use returns to the pool, or until a ConnectionWaitTimeoutException error displays. For example, if the Max Connections value is set to `5`, and there are 5 physical connections in use, the pool manager waits for the amount of time specified in Connection timeout for a physical connection to become free.

Knowing the number of connection pools that can potentially request connections from the backend, such as a DB2 database or a CICS server, helps you determine a value for the Maximum Connections property.

For multiple stand-alone application servers that use the same data source configuration, or J2C connection factory configuration, a separate physical connection pool exists for each server. If you clone these same application servers, WebSphere Application Server (base) implements a separate connection pool for each clone.

All of these connection pools correspond to the same data source or connection factory configuration. Therefore all of these connection pools can potentially request connections from the same backend resource, at the same time. The single Maximum Connections value that you set on this console panel applies to every one of these connection pools. Consequently, setting a high Maximum Connections value can result in a load of connection requests that overwhelms your backend resource.

Data type	Integer
Default	10
Range	0 to maximum integer If Max Connections is set to 0, the Connection timeout value is ignored.

Tip: For better performance, set the value for the connection pool lower than the value for the maximum thread pool connections of the web container. To configure this setting click **Servers** > **Server types**

> **WebSphere application servers** > ***server*** > **Thread Pools**, and modify the web container property. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

You can use the Tivoli® Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the processor load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Minimum connections:

Specifies the minimum number of physical connections to maintain.

If the size of the connection pool is at or below the minimum connection pool size, the Unused timeout thread does not discard physical connections. However, the pool does not create connections solely to ensure that the minimum connection pool size is maintained. Also, if you set a value for Aged timeout, connections with an expired age are discarded, regardless of the minimum pool size setting.

For example, if the Minimum Connections value is set to 3, and one physical connection is created, the Unused timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the Minimum Connections setting.

Data type	Integer
Default	1
Range	0 to max int

Reap time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if Reap Time is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the Unused timeout and Aged timeout settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused timeout and Aged timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused timeout, until it reaches the number of connections specified in Minimum Connections. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread, set Reap Time to `0`, or set both Unused timeout and Aged timeout to `0`. The recommended way to disable the pool maintenance thread is to set Reap Time to `0`, and Unused timeout and Aged timeout are ignored. However, if Unused timeout and Aged timeout are set to `0`, the pool maintenance thread runs, but only physical connections that timeout, because of non-zero timeout values, are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused timeout value higher than the Reap timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections exceeds the Minimum Connections setting. For example, if the unused timeout value is set to `120`, and the pool maintenance thread is enabled (Reap Time is not `0`), any physical connection that remains unused for 2 minutes is discarded.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting Aged timeout to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged timeout value higher than the Reap timeout value for optimal performance.

For example, if the Aged timeout value is set to `1200`, and the Reap Time value is not `0`, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. The only exception is if the connection is involved in a transaction when the aged timeout is reached, the application server will not discard the connection until after the transaction is completed and the connection is closed.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**.

Data type	String

Defaults

- EntirePool for J2C connection factories and JMS-related connection factories
- EntirePool for WebSphere Version 4.0 data sources
- EntirePool for current version data sources that you create through the administrative console
- EntirePool for current version data sources that you script through wsadmin AdminConfig commands, starting JDBC templates that are built into WebSphere Application Server. For information about the command createUsingTemplate, see the topic, Commands for the AdminConfig object.
- FailingConnectionOnly for data sources that you script in wsadmin tool without starting JDBC templates

:

Range

EntirePool
All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and issues a stale connection Exception during the next operation on that connection. Subsequent getConnection() requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this closure is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice.

FailingConnectionOnly
Only the connection that caused the stale connection exception is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a possibility that the next getConnection() request from the application can return a connection from the pool that is also stale. The result is more stale connection exceptions.

The connection pretest function attempts to insulate an application from pooled connections that are not valid. When a backend resource, such as a database, goes down, pooled connections that are not valid might exist in the free pool. This is especially true when the purge policy is failingConnectionOnly; in this case, the failing connection is removed from the pool. Depending on the failure, the remaining connections in the pool might not be valid.

Connection pool advanced settings

Use this page to specify connection pooling related settings.

This administrative console page is common to a range of resource types: for example, JDBC data sources and JMS queue connection factories. To view this page, the path depends on the type of

resource, but generally you select an instance of the resource provider, then an instance of the resource type, then click **Connection pool properties** > **Advanced connection pool properties**.

For example, click:

- **Resources** > **JDBC** > **JDBC providers** > ***JDBC_provider*** > **Data sources** > ***data_source*** > **Connection pool properties** > **Advanced connection pool properties**
- **Resources** > **JMS** > **JMS provider** > **Default messaging** > **Queue connection factory** > ***JMS_queue_connection_factory*** > **Connection pool properties** > **Advanced connection pool properties**.

The Connection Pool Partition Support creates buckets and hashes on the buckets for optimizing the connection pool for getConnection method requests. The number of shared partitions, the number of free pool partitions, and the free pool distribution table size are properties that are related to reducing the time a thread must wait for a synchronization lock. On systems with a single processor, these values do not make a difference. On systems with multiple processors, these settings can reduce the performance cost that is associated with managing multiple threads.

When the default values are used, which means that the partitions are set to 0, the connection pool automatically selects the best values. The ability to change the default values is primarily provided for connection pools that exceed 500 maximum connections. When the connection pool exceeds 500 maximum connections, the formula that is used for auto-tuning the connection pool might create large objects whose size you might want to reduce. Performance might be reduced by reducing the partition size. However, that impact is normal when you weigh memory versus performance.

Number of shared partitions:

Specifies the number of partitions that are created in each of the shared pools.

Partition support is always enabled. The default values of 0 should be used to enable the connection pool to pick the best values for performance. In some cases where large multiprocessor systems are used, adjusting the partition support properties might help performance.

Data type	integer
Default value	0
Range	0 to max int

Number of free pool partitions:

Specifies the number of partitions that are created in each of the free pools.

Data type	integer
Default value	0
Range	0 to max int

Free pool distribution table size:

Determines the distribution of Subject and CRI hash values in the table that indexes connection usage data.

These hash values are used to match connection request credentials with the connections. A free pool distribution table size larger than 1 can yield more efficient distribution of hash values, to help minimize search collisions within the table. Fewer collisions can result in faster retrieval of a connection that matches a request. Use a larger value for free pool distribution table size if your resource receives many incoming requests with varying credentials. Smaller values (1) should be used if the same credentials apply to all incoming requests for the resource. The value of 0 means random distribution.

Data type	integer
Default value	0
Range	0 to max int

Surge threshold:

Specifies the number of connections created before surge protection is activated.

Surge protection is designed to prevent overloading of a data source when too many connections are created at the same time. Surge protection is controlled by two properties, *surge threshold* and *surge creation interval*.

The surge threshold property specifies the number of connections created before surge protection is activated. After you reach the specified number of connections, you enter *surge mode*.

The surge creation interval property specifies the amount of time, in seconds, between the creation of connections when in surge mode.

For example, assume the follow settings:
- maxConnections = 50
- surgeThreshold = 10
- surgeCreationInterval = 30 seconds

If the connection pool receives 15 connection requests, 10 connections are created at about the same time. The 11th connection is created 30 seconds after the first 10 connections. The 12th connection is created 30 seconds after the 11th connection. Connections continue to be created every 30 seconds until there are no more new connections needed or you reach the maxConnections value.

Surge connection support starts if the surge threshold is > -1 and the surge creation interval is > 0. The surge threshold property has a default value of -1, which indicates that it is turned off.

Data type	integer
Default value	-1
Range	-1 to max int

Surge creation interval:

Specifies the amount of time between connection creates when you are in surge protection mode.

When the number of connections specified for the surge threshold property is reached, the surge creation interval property dictates how much time each new connection request must wait before fulfillment.

Restriction: Surge protection does not work for a connection pool that is managed through an activation specification that coordinates with a JMS queue connection factory and default messaging provider. To control incoming connections for JMS calls such as **onMessage**, see the help topic for the administrative console page **JMS** > **Activation specification** > *activation_specification_name*.

Data type	integer
Default value	0
Range	0 to max int

Stuck timer interval:

A *stuck* connection is an active connection that is not responding or returning to the connection pool. If the pool is stuck (you have reached the stuck threshold), a resource exception is given to all new connection requests until the pool is unstuck. The stuck timer interval property is the interval for the timer, for example, how often the connection pool checks for stuck connections. The default value is 0 seconds.

If an attempt to change the stuck time, stuck timer interval, or stuck threshold properties using the wsadmin scripting tool fails, an `IllegalState` exception occurs. The pool cannot have any active requests or active connections during this request. For the stuck connection support to start, the stuck time and the stuck threshold property values must be greater than 0 and maximum connections must be greater than 0.

Also, the stuck timer interval, if it is set, must be less than the stuck time value. In fact, it is suggested that the stuck timer interval be one-quarter to one-sixth the value of stuck time so that the connection pool checks for stuck connections 4 - 6 times before a connection is declared stuck. This interval check reduces the likelihood of false positives.

Data type	integer
Default value	0
Range	0 to max int

Stuck time:

A *stuck* connection is an active connection that is not responding or returning to the connection pool. If the pool is stuck (you have reached the stuck threshold), a resource exception is given to all new connection requests until the pool is unstuck. The stuck time property is the interval, in seconds, allowed for a single active connection to be in use to the backend resource before it is considered to be stuck.

Data type	integer
Default value	0
Range	0 to max int

Stuck threshold:

A *stuck* connection is an active connection that is not responding or returning to the connection pool. If the pool is stuck (you have reached the stuck threshold), a resource exception is given to all new connection requests until the pool is unstuck. An application can explicitly catch this exception and continue processing. The pool continues to periodically check for stuck connections when the number of stuck connections is past the threshold. If the number of stuck connections drops below the stuck threshold, the pool detects this during its periodic checks and enables the pool to begin servicing requests again. The stuck threshold is the number of connections that must be considered stuck for the pool to be in stuck mode.

Data type	integer
Default value	0
Range	0 to max int

Connection pool (Version 4) settings

Use this page to create a connection pool for a Version 4.0 data source.

You can access this administrative console page in one of two ways:

- **Resources** > **JDBC** > **JDBC Providers** > *JDBC_provider* > **Data sources (WebSphere Application Server V4)** > *data_source* > **Connection pool properties (version 4)**
- **Resources** > **JDBC** > **Data sources (WebSphere Application Server V4)** > *data_source* > **Connection pool properties (version 4)**

Scope: Resources such as JDBC providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates which are defined at more general scopes.

Note that no matter what the scope of a defined resource, the resource's properties only apply at an individual server level. For example, if you define the scope of a data source at the cell level, all users in that cell can look up and use that data source, which is unique within that cell. However, resource property settings are local to each server in the cell. For example, if you define *max connections* to 10, then each server in that cell can have 10 connections.

When resources are created, they are always created into the current scope selected in the panel. To view resources in other scopes, specify a different node or server in the scope selection form.

For general information, see *Administrative console scope settings* in the Related Reference section.

Data type String

Minimum pool size:

Specifies the minimum number of connections to maintain in the pool.

The minimum pool size can affect the performance of an application. Smaller pools require less overhead when the demand is low because fewer connections are held open to the database. When the demand is high, the first applications experience a slow response because new connections are created if all others in the pool are in use.

Data type Integer
Default 1
Range Any non-negative integer.

Maximum pool size:

Specifies the maximum number of connections to maintain in the pool.

If the maximum number of connections is reached and all connections are in use, additional requests for a connection wait up to the number of seconds specified as the connection timeout. The maximum pool size can affect the performance of an application. Larger pools require more overhead when demand is high because there are more connections open to the database at peak demand. These connections persist until idled out of the pool. If the maximum value is smaller, longer wait times or possible connection timeout errors during peak times can occur. Ensure that the database can support the maximum number of connections in the application server, in addition to any load that it has outside of the application server.

Data type Integer
Default 10
Range Any positive integer

Connection timeout:

Specifies the maximum number of seconds an application waits for a connection from the pool before timing out and triggering a `ConnectionWaitTimeout` exception. WebSphere Application Server acts on this value only if you set the maximum pool size property, in which case the number of maximum connections serves as a trigger for enforcing the wait timeout property.

Data type Integer

Units	Seconds
Default	180
Range	Any non-negative integer

Setting this value to 0 disables the connection timeout.

If you accept the default value, Application Server issues the ResourceAllocation exception immediately after the pool manager indicates that the maximum number of connections are in use. If you disable connection timeout, Application Server does not issue an exception. Instead, the pool manager queues subsequent connection requests until it can allocate a connection.

Idle timeout:

Specifies the maximum number of seconds that an idle (unallocated) connection can remain in the pool before being removed to free resources.

Connections need to idle out of the pool because keeping connections open to the database can cause database memory problems. However, not all connections are idled out of the pool, even if they are older than the Idle Timeout setting. A connection is not idled if removing the connection would cause the pool to shrink below its minimum size. Setting this value to 0 disables the idle timeout.

Data type	Integer
Units	Seconds
Default	1800
Range	Any non-negative integer

Orphan timeout:

Specifies the maximum number of seconds that an application can hold a connection without using it before the connection returns to the pool

If there is no activity on an allocated connection for longer than the Orphan Timeout setting, the connection is marked for orphaning. After another Orphan Timeout number of seconds, if the connection still has no activity, the connection returns to the pool. If the application tries to use the connection again, it is issued a stale connection exception. Connections that are enlisted in a transaction are not orphaned. Setting this value to 0 disables the orphan timeout.

Data type	Integer
Units	Seconds
Default	1800
Range	Any non-negative integer

Statement cache size:

Specifies the number of cached prepared statements to keep per connection.

The largest value you would need to set your cache size to if you do not want any cache discards is determined as follows: for each application that uses this data source on a particular server, add up the number of unique prepared statements (as determined by the *sql* string, concurrency, and the scroll type). This is the maximum number of possible prepared statements that can be cached on a given connection over the life of the server. Setting the cache size to this value means you never have cache discards. This provides better performance. However, because of potential resource limitations, this might not always be possible.

Data type	Integer
Default	10
Range	Any non-negative integer

Disable auto connection cleanup:

Specifies whether the connection pooling software automatically closes connections from the data source at the end of a transaction. Set this property if you want to maintain and reuse the same connection across multiple transactions.

The default is *false*, which indicates that when a transaction completes, the application server closes the connection and returns it to the pool. Any use of the connection after the transaction has ended results in a stale connection exception, because the connection is closed and has returned to the pool. This mechanism ensures that connections are not held indefinitely by the application. If the value is set to true, the connection is not returned to the pool at the end of a transaction. In this case, the application must return the connection to the pool by calling the close() method. If the application does not close the connection, the pool can run out of connections for other applications to use.

Data type	Boolean (check box)
Default	False (clear)

J2C Connection Factories collection

Use this page to view Java 2 Connector (J2C) connection factories, which represent sets of connection configuration values.

Application components such as enterprise beans have resource reference descriptors that refer to the connection factory, not the resource adapter. The connection factory is really a configuration properties list holder. In addition to the arbitrary set of configuration properties defined by the vendor of the resource adapter, there are several standard configuration properties that apply to the connection factory. These standard properties are used by the Java 2 Connectors connection pool manager in the application server run time and are not known by the vendor-supplied resource adapter code.

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Adapters** > **J2C connection factories**
- **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **J2C connection factories**

Name:

Specifies the display name of a connection factory.

Data type	String

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name of this connection factory.

Data type	String

Scope:

Specifies the scope of the connection factory. Only applications that are installed within this scope can use this connection factory.

Provider:

Specifies the resource adapter that WebSphere Application Server uses for this connection factory.

Description:

Specifies a text description of this connection factory.

Data type String

Connection factory interface:

Specifies the fully qualified name of the interface that provides the implementation class for the connection factory.

Category:

Specifies a string that you can use to classify or group this connection factory.

Data type String

J2C connection factories settings:

Use this panel to specify settings for a connection factory.

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Adapters** > **J2C connection factories** > ***J2C_connection_factory***
- **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **J2C connection factories** > ***J2C_connection_factory***

Scope:

Specifies the scope of the resource adapter that connects applications to an enterprise information system (EIS) through this connection factory. Only applications that are installed within this scope can use this connection factory.

Provider:

Specifies the resource adapter that WebSphere Application Server uses for this connection factory.

Provider can be set only when you create a new connection factory. The list shows all of the existing resource adapters that are defined at the relevant scope. Select one from the list if you want to use an existing resource adapter as Provider.

Create new provider:

Provides the option of configuring a new resource adapter for the new connection factory.

Create New Provider is displayed only when you create, rather than edit, a connection factory.

Clicking **Create New Provider** triggers the console to display the resource adapter configuration page, where you create a new adapter. After you click **OK** to save your settings, you see the connection factory collection page. Click **New** to define a new connection factory for use with the new resource adapter; the console now displays a configuration page that lists the resource adapter as the new connection factory Provider.

Name:

Specifies the name of this connection factory.

This is a required property.

Data type String

JNDI name:

Specifies the JNDI name of this connection factory.

For example, the name could be *eis/myECIConnection*.

After you set this value, save it and restart the server. You can see this string when you run the *dumpNameSpace* tool. This is a required property. If you do not specify a JNDI name, it is completed by default using the Name field.

Data type String
Default eis/*display name*

Important: Adhere to the following requirements for JNDI names:

- Do not assign duplicate JNDI names across different resource types (such as data sources versus J2C connection factories or JMS connection factories).
- Do not assign duplicate JNDI names for multiple resources of the same type in the same scope.

Description:

Specifies a text description of this connection factory.

Data type String

Connection factory interface:

Specifies the fully qualified name of the Connection Factory Interfaces supported by the resource adapter.

This is a required property. For new objects, the list of available classes is provided by the resource adapter in a drop-down list. After you create the connection factory, the field is a read only text field.

Data type Drop-down list or text

Category:

Specifies a string that you can use to classify or group this connection factory.

Data type String

Authentication alias for XA recovery:

Specifies the authentication alias that is used during XA recovery processing. If this alias name is changed after a server failure, the subsequent XA recovery processing uses the original setting that was in effect before the failure.

Select an alias from the list.

To define a new alias that is not displayed in the list:

1. Click **Apply**. Under Related Items, you now see a listing for Java Platform, Enterprise Edition (Java EE) Connector Architecture (J2C) authentication data entries.
2. Click **JAAS - J2C authentication data**.
3. Click **New**.
4. Define an alias.
5. Click **OK** and **Save**. The console now displays an alias collection page that lists all configured aliases. Above the table, this page also displays the name of your connection factory in the breadcrumb path.
6. Click the name of your J2C connection factory to return to the configuration panel for the connection factory that you are creating.
7. Select the new alias in the container-managed authentication alias list.
8. Click **Apply**.

If the resource adapter does not support XA transactions, this field is not displayed. The default value comes from the selected alias for application authentication, if specified.

If you have defined multiple security domains and multiple authentication aliases in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. Security domains allow you to isolate authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that is able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them.

The browse button is only accessible if at least one security domain is defined and assigned a scope that is applicable to the resource that is being edited. Additionally, that security domain must contain at least one JAAS J2C Authentication alias.

Data type Drop-down list

Component-managed authentication alias:

Specifies authentication data for component-managed signon to the resource.

Select an alias from the list.

To define a new alias that is not displayed in the list:

1. Click **Apply**. Under Related Items, you now see a listing for Java Platform, Enterprise Edition (Java EE) Connector Architecture (J2C) authentication data entries.
2. Click **JAAS - J2C authentication data**.
3. Click **New**.
4. Define an alias.
5. Click **OK** and **Save**. The console now displays an alias collection page that lists all configured aliases. Above the table, this page also displays the name of your connection factory in the breadcrumb path.
6. Click the name of your J2C connection factory to return to the configuration panel for the connection factory that you are creating.
7. Select the new alias in the container-managed authentication alias list.
8. Click **Apply**.

If you have defined multiple security domains and multiple authentication aliases in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. Security domains allow you to isolate authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that is able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them.

The browse button is only accessible if at least one security domain is defined and assigned a scope that is applicable to the resource that is being edited. Additionally, that security domain must contain at least one JAAS J2C Authentication alias.

Data type List

The alias that you configure for component-managed authentication does not apply to all clients that must access the secured resource. External Java clients with Java Naming and Directory Interface (JNDI) access can look up a Java 2 Connector (J2C) resource such as a data source or Java Message Service (JMS) queue. However, they are not permitted to take advantage of the component-managed authentication alias defined on the resource. This alias is the default value that is used when the getConnection() method does not specify any authentication data, like *user* and *password*, or a value for ConnectionSpec. If an external client needs a connection, it must assume responsibility for the authentication by passing it through arguments on the getConnection() call.

Mapping-configuration alias:

Specifies the authentication alias for the Java Authentication and Authorization Service (JAAS) mapping configuration that is used by this connection factory.

The **DefaultPrincipalMapping** JAAS configuration maps the authentication alias to the user ID and password. You can define and use other mapping configurations.

Note: Some mapping-configuration aliases do not use container-managed authentication aliases, so you cannot select a container-managed authentication alias if one of those mapping-configuration aliases is selected.

Data type Pick-list

Container-managed authentication alias:

Specifies authentication data, which is a JAAS - J2C authentication data entry, for container-managed signon to the resource. This setting can be disabled depending on the value that is selected for the Mapping-configuration alias setting.

Select an alias from the list.

To define a new alias that is not displayed in the list:

1. Click **Apply**. Under Related Items, you now see a listing for Java Platform, Enterprise Edition (Java EE) Connector Architecture (J2C) authentication data entries.
2. Click **JAAS - J2C authentication data**.
3. Click **New**.
4. Define an alias.
5. Click **OK** and **Save**. The console now displays an alias collection page that lists all configured aliases. Above the table, this page also displays the name of your connection factory in the breadcrumb path.

6. Click the name of your J2C connection factory to return to the configuration panel for the connection factory that you are creating.
7. Select the new alias in the container-managed authentication alias list.
8. Click **Apply**.

If you have defined multiple security domains and multiple authentication aliases in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. Security domains allow you to isolate authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that are able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them.

The browse button is only accessible if at least one security domain is defined and assigned a scope that is applicable to the resource that is being edited. Additionally, that security domain must contain as least one JAAS J2C Authentication alias.

Data type Pick-list

Authentication preference:

Specifies the authentication mechanisms defined for this connection factory.

This setting specifies which of the authentication mechanisms defined for the corresponding resource adapter applies to this connection factory. Common values, depending on the capabilities of the resource adapter, are: `KERBEROS`, `BASIC_PASSWORD`, and `None`.

If `None` is chosen, the application component is expected to manage authentication (<res-auth>Application</res-auth>). In this case, the user ID and password are taken from one of the following:
- The component-managed authentication alias
- UserName, Password Custom Properties
- Strings passed on the getConnection method

For example, if two authentication mechanism entries are defined for a resource adapter in the *ra.xml* document:
- <authentication-mechanism-type>BasicPassword</authentication-mechanism-type>
- <authentication-mechanism-type>Kerbv5</authentication-mechanism-type>

The authentication preference specifies the mechanism to use for container-managed authentication. An exception is issued during server startup if a mechanism that is not supported by the resource adapter is selected.

Data type Pick-list
Default BASIC_PASSWORD

J2C Connection Factory advanced settings:

Use this page to specify settings for a Java 2 Connector (J2C) connection factory.

You can access this administrative console page in one of two ways:
- **Resources** > **Resource Adapters** > **J2C connection factories** > ***J2C_connection_factory*** > **Advanced connection factory properties**
- **Resources** > **Resource Adapters** > ***resource_adapter*** > **J2C connection factories** > ***J2C_connection_factory*** > **Advanced connection factory properties**

Log missing transaction contexts:

Specifies whether the container logs that there is a missing transaction context when a connection is obtained.

Data type	Boolean
Default	True (enabled)

Cached handles:

Specifies whether cached handles are tracked by the container. Cached handles are handles that are held in `inst vars` in a bean.

Data type	Boolean
Default	False (clear)

Resource workload routing:

Use this topic to learn how to enable resource routing in your environment.

Note: Resource workload routing includes data source and connection factory fail over and subsequent fail back from a predefined alternate or backup resource. This function enables applications to easily recover from resource outages, such as database failures, without requiring you to embed alternate resource and configuration information. You can tailor the resource fail over and failback flexible configuration options to meet your environment-specific and application needs.

Configure an alternate resource

A data source and connection factory can fail over and fail back automatically when a specified or default failure threshold value is reached. When fail over occurs, the application switches from using the primary resource to using the alternate resource. Fail back occurs when the application switches back from the alternate resource to the primary resource.

The alternate resource is created the same way that other connection factories or data sources are created. The alternate resource configuration should mirror the primary resource configuration. For example, the alternate resource security configuration and the primary resource configuration, with respect to the application and resources, should mirror each other so that the application and database can access the required data. After the alternate resource is created, you can change the database values that are necessary for the alternate resource backend configuration. If the alternate resource is not compatible, it is likely that fail over will fail. If the resources are not compatible, the following errors might occur: tables or fields that do not exist, an expected record does not exist and unexpected resource errors exist. As a test option, before the alternate resource is configured to the primary data source, you can test the application by changing the JNDI name in the application from the primary JNDI name to the alternate JNDI name.

When the primary resource is used, the alternate resource is paused. Before the alternate resource is paused, the alternate resource can be available before the primary is used. Using the alternate before it is paused is not recommended unless there is a special reason the alternate needs to be accessed before the primary resource. An example of a special reason is testing the application for compatability.

An alternate resource cannot be used as a primary. When using the fail over feature with non-relational resource adapters that have back-ends that also support fail over, you must verify that fail over is not configured for these back-ends. Fail over works with non-relational resource adapters that have a ManagedConnection object that implements a testConnection method. The testConnection method is used to ping the alternate and primary resources for success before re-establishing a connection to the currently available resource. If the resource adapter does not implement a testConnection method or testConnection throws a `javax.resource.NotSupportedException` error, the fail over feature is disabled.

For resource adapters that do not meet the requirement for testConnection, partial fail over can be used. You must manually fail back using Mbeans to the primary resource when the primary resource is available. Partial fail over can be enabled by setting the property, enablePartialResourceAdapterFailoverSupport to `true`.

You are encouraged to test the suitability of this feature with your system environment and resources before enabling fail over support.

Mbean operations properties

failOverToAlternateResource

Values: `none`, `hold` or `automated`; default is `hold` without automated fail back.

Description: Manual fail over to alternate resource. This action is issued on the primary resource.

failBackToPrimaryResource

Values: `none`, `hold` or `automated`; default is automated with automated fail over.

Description: Manual fail back to primary resource. This action is issued on the primary resource.

Mbean attribute properties

resourceFailOver

Values: boolean

Description: False - Disables resource fail over. True - Enables resource fail over. This action is issued on the primary resource.

resourceFailBack

Values: boolean

Description: False - Disables resource fail back. True - Enables resource fail back. This action is issued on the primary resource.

populateAlternateResource

Values: boolean

Description: False - Disables populate alternate resource. True - Enables populate alternate resource. This action is issued on the alternate resource.

Custom properties

All properties for this feature must be created as new custom properties on the connection pool for a particular data source or connection factory. In the administrative console, navigate to the data source or connection factory that notification is to be enabled for. Click the **Connection pool properties** link. On the **Pool Properties** panel, click **Connection pool custom properties** link. The **Custom properties** panel for the resource connection pool displays. Click **New** to create the custom properties described as follows:

failureThreshold

Values: Must be an integer and > `0`.

Description: The failureThreshold property is only read if the failureNotificationActionCode or alternateResourceJNDIName property is set to valid value. If the failureThreshold property is not set or is set to an invalid number the default value of `5` is used. The integer value specified for the failureThreshold is the number of consecutive resource exceptions that must occur for a particular resource before notification is sent or failover occurs.

The following is an example of how this value works: If the failureThreshold property is set to `5` for data source B, then data source B must get five consecutive resource exceptions while attempting to establish connections, with no successful attempts in-between these failures, before notification is sent or the resource can fail over. An attempt to send the notification or fail over is made after five consecutive resource exceptions occur. However, in a multi-threaded environment, after the failure threshold value is reached, the timing of the notification or fail over might occur after additional resource exceptions.

Attention: Resource exception counters are not shared between resources. Resource exceptions must be consecutive to reach failure threshold.

alternateResourceJNDIName

Values: String value containing a direct JNDI name.

Description: An alternate connection factory or data source resource should be like the primary resource. Provide the JNDI name of the alternate resource to enable the fail over feature.

Advanced fail over properties

resourceAvailabilityTestRetryInterval

Values: int value, default is `10`.

Description: The test connection interval by default is `10` seconds. Every 10 seconds, the test connection thread attempts to test the primary resource. Depending on system resources, this value can be change from `1` - `MAXINIT` seconds.

enablePartialResourceAdapterFailoverSupport

Values: boolean value, default is `false`.

Description: If this value is `true`, fail over to the alternate resource occurs, but fail back to the primary is manual. This property can be set if the resource adapter being used does not meet the requirements for connection fail over, for example, it does not have testConnection implemented or it throws a not supported exception.

disableResourceFailOver

Values: boolean value, default is `false`.

Description: If this value is `true`, automatic fail over does not occur.

disableResourceFailBack

Values: boolean value, default is `false`.

Description: If this value is `true`, automatic fail back does not occur.

populateAlternateResource

Values: boolean value, default is `false`.

Description: If this value is `true`, the alternate resource is populated with connections to maximum connections. Every attempt is made to keep the alternate resource at maximum connections. If the database goes down for the alternate resource, the stale connections are removed and the populate thread repopulates the alternate resource when the database is available. You can dynamically enable and disable the populate alternate resource using the Mbean operations, disablePopulateAlternateResource and enablePopulateAlternateResource.

Attention: You might see performance gains during a failover with populate alternate resource enabled, but, the cost is high to keep the alternate resource populated. Most of the cost is in keeping twice the number of connections typically required for one data source. Because connections are large objects, keeping the connections uses additional memory resource on the server and extra resource on the database.

Data source resource definition in applications:

In support of the Java Enterprise Edition (Java EE) 6 specification, applications can define data sources in annotations or in the deployment descriptor. This topic reviews similarities and compatibility with WebSphere Application Server data sources defined at the server, node, cluster, or cell level. Optional features in data source definition are also discussed.

Standard properties for data source definition

Table 18. Standard properties for data source definition. Use this table to learn about standard properties for data source definitions

Annotation element	Descriptor element	Comments
name	name	JNDI name for the data source. The name must be in one of the java:global, java:app, java:module, or java:comp name spaces.
className	class-name	Fully qualified class name from the JDBC driver that implements javax.sql.XADataSource, javax.sql.ConnectionPoolDataSource, or javax.sql.DataSource.
databaseName	database-name	Value is supplied to JDBC driver.
description	description	Value is supplied to DataSource MBean.
initialPoolSize	initial-pool-size	Value of property is ignored. In WebSphere Application Server, initial pool size is always `0`.
isolationLevel	isolation-level	Equivalent to WebSphere Application Server data source custom property, webSphereDefaultIsolationLevel. This is a default transaction isolation level for new connections.
loginTimeout	login-timeout	Value is supplied to JDBC driver.
maxIdleTime	max-idle-time	Equivalent to WebSphere Application Server connection pooling property, unusedTimeout. This property is ignored in the client container where connection pooling is not provided.

Table 18. Standard properties for data source definition (continued). Use this table to learn about standard properties for data source definitions

Annotation element	Descriptor element	Comments
maxPoolSize	max-pool-size	Equivalent to WebSphere Application Server connection pooling property, maxConnections. This property is ignored in the client container where connection pooling is not provided.
maxStatements	max-statements	Defines the maximum number of statements for the connection pool. In WebSphere Application Server, each pooled connection has its own statement cache. Consequently, maxStatements is divided (equally, rounding down) between the maxPoolSize for the pool. If maxPoolSize is unlimited, then statement pooling is disabled.
minPoolSize	min-pool-size	Equivalent to WebSphere Application Server connection pooling property, minConnections. This property is ignored in the client container where connection pooling is not provided.
password	password	Default password for connection requests that do not specify a password. Consider using an authentication alias instead of hard-coding the user name and password into the application.
portNumber	port-number	Value is supplied to JDBC driver.
serverName	server-name	Value is supplied to JDBC driver.
transactional	transactional	In WebSphere Application Server, the property, transactional, controls whether the connection is enlisted in JTA transactions. When transactional=false, connections are not enlisted in JTA transactions, but you can still run transactions against the database using autocommit=true or connection.commit/rollback with autocommit=false. Equivalent to the opposite of the WebSphere Application Server data source custom property, nonTransactionalDataSource.
url	url	Value is supplied to the JDBC driver unless any of the following are also specified: databaseName, serverName, portNumber.
user	user	Default user name for connection requests that do not specify a user name. Consider using an authentication alias instead of hard-coding the user name and password into the application.

Table 18. Standard properties for data source definition (continued). Use this table to learn about standard properties for data source definitions

Annotation element	Descriptor element	Comments

Vendor properties and custom properties

JDBC driver vendor properties can be included in the data source definition. Most of the WebSphere Application Server custom properties can also be included in the data source definition.

With annotations, this is done through the properties element; for example,

```
@DataSourceDefinition
(
  name="java:app/env/myDataSource",
  className="org.apache.derby.jdbc.EmbeddedXADataSource40",
  databaseName="myDB",
  properties=
  {
    // Vendor properties for Derby Embedded JDBC driver:
    "createDatabase=create",
    "connectionAttributes=upgrade=true",

    // Custom properties for WebSphere Application Server:
    "connectionTimeout=60",
    "dataStoreHelperClass=com.ibm.websphere.rsadapter.DerbyDataStoreHelper",
    "validateNewConnection=true",
    "validateNewConnectionRetryCount=5"
  },
  serverName=""
)
```

The following example illustrates how the data source definition is included in the deployment descriptor:

```
<data-source>
 <name>java:app/env/myDataSource</name>
 <class-name>org.apache.derby.jdbc.EmbeddedXADataSource40</class-name>
 <database-name>myDB</database-name>
 <property><name>createDatabase</name><value>create</value></property>
 <property><name>connectionAttributes</name><value>upgrade=true</value></property>
 <property><name>connectionTimeout</name><value>60</value></property>
 <property><name>dataStoreHelperClass</name><value>com.ibm.websphere.rsadapter.DerbyDataStoreHelper</value></property>
 <property><name>validateNewConnection</name><value>true</value></property>
 <property><name>validateNewConnectionRetryCount</name><value>5</value></property>
 <server-name/>
</data-source>
```

The following is a list of WebSphere Application Server custom properties that can be configured in this manner:

- Connection pooling properties:
  - agedTimeout
  - authDataAlias
  - authMechanismPreference
  - connectionTimeout
  - defaultConnectionTypeOverride
  - globalConnectionTypeOverride
  - mappingConfigAlias
  - purgePolicy

  - reapTime
  - stuckThreshold
  - stuckTime
  - stuckTimerTime
  - surgeCreationInterval
  - surgeThreshold
  - testConnection
  - testConnectionInterval
  - XA_RECOVERY_AUTH_ALIAS
- Data source custom properties:
  - beginTranForResultSetScrollingAPIs
  - beginTranForVendorAPIs
  - connectionSharing
  - enableMultithreadedAccessDetection
  - errorDetectionModel
  - freeResourcesOnClose
  - oracleRACXARecoveryDelay (Oracle only)
  - preTestSQLString
  - userDefinedErrorMap
  - validateNewConnection
  - validateNewConnectionRetryCount
  - validateNewConnectionRetryInterval
  - validateNewConnectionTimeout

Resource references

It is recommended for applications to always use resource references when accessing data sources, which makes it easier for the deployer to override.

Connection sharing

By default, for data source definition, a connection request can share an existing in-use connection if it matches the originally requested settings for that connection (connectionSharing=MatchOriginalRequest). Alternately, connection sharing can be done by matching the connection request against the current state of the connection (connectionSharing=MatchCurrentState).

Life cycle

The life cycle of a data source definition is tied to the life cycle of the applications that define it. Consequently, you can update your application to change the data source definition without needing to restart the server. If multiple applications include the same data source definition, for example, both data source definitions have identical java:global names, identical set of properties configured, and identical values for properties, then all of the applications must be uninstalled before updating the data source definition and reinstalling the applications.

Conflicts between data source definitions

Applications, modules, and components should take care not to define data sources with the same java:global name as another application, because that process makes it impossible for the applications to coexist. Within an application, modules and components should take care not to define data sources with

the same java:app name as another module or component. The conflict causes the application installation to fail. Within a module, components should take care not to define data sources with the same java:module name as other components. The conflict causes the application installation to fail. Within the web module, components should take care not to define data sources with the same java:comp name as other components. The conflict causes the application installation to fail.

Bean validation in RAR modules:

WebSphere Application Server validates resource adapter archive (RAR) JavaBeans constraints in compliance with the Java Connector Architecture (JCA) version 1.6 specification.

Resource adapters can specify the validation requirements of configuration properties to the Application Server through annotations in the source code of the resource adapter, constraint specifications in a resource adapter validation descriptor, or a mixture of both. In specifying these constraints, resource adapters can use the built-in bean validation constraints supplied with the Application Server, custom bean validation constraints supplied either by the application developer or a third party, or a mixture of both. Resource adapter developers can apply constraints to the fields and JavaBeans-compliant properties of the following JCA types:

- ResourceAdapter
- ManagedConnectionFactory
- ActivationSpec
- AdministeredObject

At run time, the application server creates instances of bean types declared by the resource adapter. Each instance is validated immediately upon setting its configuration properties, before placing the instance into service.

When validating a RAR bean, the Application Server creates an instance of a validator factory according to the bean validation deployment descriptor discovered by the Application Server. A validator instance is then obtained from the factory and used to validate the bean instance.

If validation fails, the Application Server throws a constraint violation exception and reports all violations to the system log. The effects of the exception for each RAR bean type and problem determination information are documented in the topic, Troubleshooting bean validation in RAR modules.

Note: The Bean Validation specification requires that no more than one `validation.xml` is visible on the class path. This requirement is violated whenever two or more stand-alone RARs provide a validation descriptor. See the section, "RAR bean validation descriptor" in this topic, for more information. When more than one `validation.xml` is visible to the Application Server class loaders, the Application Server or application modules might fail to acquire the default ValidatorFactory and subsequently cannot perform bean validation. For example, the server cannot validate beans of a RAR embedded in an application whenever the embedded RAR lacks a validation configuration, and two or more stand-alone RARs provide configurations. To avoid trouble, install stand-alone RARs that provide a bean validation descriptor as isolated whenever possible.

Built-in constraint annotations

Note: Use built-in constraint annotations to specify the range and mandatory attributes of configuration properties rather than provide custom annotations for the same purpose. The following constraints are useful, but you can use all bean validation built-in constraints. See the topic Bean validation built-in constraints for a complete list of the constraints.

- @Min

  Specifies the minimum value of the configuration property decorated with this annotation. The value must be greater than or equal to the specified minimum.

- @Max

Specifies the maximum value of the configuration property decorated with this annotation. The value must less than or equal to the specified maximum.

- @Size

  Specifies the range of values of the configuration property decorated with this annotation. The value must be greater than or equal to the specified minimum and be less than or equal to the specified maximum.

- @NotNull

  Specifies the value of the configuration property decorated with this annotation must not be null. That is, the property is required.

The following example is a RAR bean class that is decorated with built-in constraint annotations.

The value of the serverName configuration property must not be null, and the value of the instanceCount property must be at least 1 when the Application Server creates and configures an instance of the MyConnector class. Otherwise, a constraint validation exception occurs and, in the case of ResourceAdapter bean, the resource adapter fails to start. See the topic Troubleshooting bean validation in RAR modules for more information.

```
package com.my.company;

@Connector(...)
public class MyConnector implements ResourceAdapter, Serializable
{
 @ConfigProperty(type=java.lang.String.class,defaultValue="WAS")
 private String serverName;

 @NotNull()
 public String getServerName() {return serverName;}

 private Integer instanceCount = 0;

 @Min(value=1)
 public Integer getInstanceCount() {return instanceCount;}
 ...
```

RAR bean validation descriptor

Bean validation constraints can be declared through an XML descriptor supplied by a RAR module. In the simplest case, a RAR validation descriptor consists of the validation configuration declared in the validation.xml file and zero or more XML files that declare RAR bean validation constraints. Files containing constraint declarations are specified in the constraint-mapping elements of the validation configuration (validation.xml).

You must package the validation descriptor in the META-INF directory of a RAR module. Any custom constraint annotation classes that are declared in the validation descriptor must also be packaged in the RAR module.

The following example is a simple RAR validation descriptor that declares constraint metadata like the code shown in the section, "Built-in constraint annotations."

```
<?xml version="1.0" encoding="UTF-8"?>
<validation-config
  xmlns="http://jboss.org/xml/ns/javax/validation/configuration"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/configuration validation-configuration-1.0.xsd">
<constraint-mapping>META-INF/constraints.xml</constraint-mapping>
</validation-config>
```

The constraints XML file is also located in the META-INF directory and looks like the following:

```
<constraint-mappings
 xmlns="http://jboss.org/xml/ns/javax/validation/mapping"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://jboss.org/xml/ns/javax/validation/mapping validation-mapping-1.0.xsd">
<default-package>com.my.company</default-package>
<bean class="MyConnector" ignore-annotations="true">
 <field name="serverName">
  <valid/>
  <!-- @NotNull() -->
```

```
  <constraint annotation="javax.validation.constraints.NotNull">
   <message>Value is not null</message>
  </constraint>
 </field>
 <field name="instanceCount">
  <valid/>
  <!-- @Min(1) -->
  <constraint annotation="javax.validation.constraints.Min">
    <message>Minimum possible value is 1</message>
    <element name="value">1</element>
   </constraint>
  </field>
 </bean>
<constraint-mapping>
```

The packaged RAR module, `MyResourceAdapter.rar`, looks like the following:

```
my/
  company/
    MyConnector.class
. . .
META-INF
  /validation.xml
  /constraints.xml
```

Third-party bean validation

WebSphere Application Server supports using different bean validation implementations. If a resource adapter requires a bean validation implementation different from the implementation that is provided by the product, and the RAR provides the bean validation implementation, you must package the JAR file that contains the bean validation implementation in the RAR module root directory.

The RAR module must also contain a single validation configuration descriptor (`validation.xml`), which can be packaged in the META-INF directory of the RAR module, or in the `META-INF/services` directory of the bean validation JAR file, but not both.

RAR bean validation configuration discovery

When validating RAR beans, the Application Server bootstraps the bean validation configuration, specific to the RAR, according to the bean validation descriptor supplied in the RAR META-INF directory. If the descriptor does not exist, the server bootstraps the configuration using the first validation descriptor discovered in the RAR class loading context, such as that supplied in a third-party bean validation that is packaged in the RAR. Finally, the server uses the default validation configuration provided by the product.

The server then creates a validator factory specific to the discovered bean validation configuration and uses this factory to create validator instances for validating the RAR bean instances. When you deploy a RAR that supplies a bean validation descriptor, you must take additional steps to ensure that the class loader that loads the RAR loads the bean validation descriptor and classes packaged in the RAR.

For an embedded RAR, after you have deployed the application that embeds the RAR, you must set the delegation mode of the application class loader to Parent-Last (Child-First). See the topic Configuring application class loaders for more information.

For a stand-alone RAR, you must install the RAR as an isolated resource provider. See the topic Resource Adapter settings for more information.

Troubleshooting bean validation in RAR modules:

RAR beans that fail validation are not placed into service. When constraint violations occur, applications encounter resource connectivity issues that are different according to the bean type and how the RAR is deployed. This topic explains how to understand, service, and prevent these known issues.

RAR bean constraint violations

WebSphere Application Server displays a constraint violation exception and reports all constraint violations to the system log when it validates RAR bean instances that violate one or more constraints. The cause of all constraint violation must be determined and resolved to restore full connectivity to the affected resource.

Problem determination starts with consulting the RAR provider documentation for the valid values of the configuration properties that are indicated in the violations. If the property values are invalid, you must reconfigure them according to the documentation and restart the resource adapter. If the adapter is embedded in an application, then restart the application to restart the adapter; if the adapter is stand-alone, then restart the application server.

If a valid configuration property value is indicated in a violation, then the constraint might be incorrectly specified for the bean, or the bean is incorrectly computing the property value. In these cases, the RAR vendor must correct the problem.

If the problem is caused by a faulty constraint definition (implementation), then the bean validation provider must correct the problem. In these cases, if the RAR is provided by IBM, or the RAR uses the bean validation implementation supplied by the Application Server, then contact IBM support to continue problem determination.

ResourceAdapter beans

ResourceAdapter beans are validated when the server starts a Java 2 Connector (J2C) resource adapter. When validation fails, the server rejects the ResourceAdapter instance and the resulting constraint violation exception causes the J2C resource adapter to fail. Applications cannot establish outbound connections to the resource, and the resource cannot deliver messages to applications. For an embedded adapter, the application that embeds the adapter fails to start. In-doubt transactions that involve the resource cannot be recovered.

The following example is a ResourceAdapter bean, MyConnector, at heap address 7efa7efa. Two validation constraints are violated. The constraint violation exception causes J2CResourceAdapter_1285109360562 to fail:

```
[9/29/10 10:51:24:125 CDT] 00000000 BeanValidatio E
 J2CA0238E: JavaBean com.my.company.adapter.MyConnector@7efa7efa failed Bean Validation due to one or more invalid
 configuration settings indicated in the following list of constraint violations:

 ConstraintViolationImpl{interpolatedMessage='The minimum size is 2', propertyPath=dataBaseName, rootBeanClass=class
    com.my.company.adapter.MyConnector, messageTemplate='The minimum size is 2'}
 ConstraintViolationImpl{interpolatedMessage='must be greater than or equal to 10', propertyPath=idleTimeout, rootBeanClass=class
    com.my.company.adapter.MyConnector, messageTemplate='{javax.validation.constraints.Min.message}'}
...
[9/29/10 10:51:24:468 CDT] 00000000 RALifeCycleMa E
 J2CA0128E: An Exception occurred while trying to start ResourceAdapter
  cells/IBM-46DF84D297BNode01Cell/nodes/IBM-46DF84D297BNode01/resources.xml#J2CResourceAdapter_1285109360562. The exception is:
com.ibm.ejs.j2c.metadata.ConstraintViolationException
 at com.ibm.ejs.j2c.metadata.BeanValidationHelper.validate(
 at com.ibm.ejs.j2c.RAWrapperImpl.createAndConfigureRA(
 at com.ibm.ejs.j2c.RAWrapperImpl.startRA(
 at com.ibm.ejs.j2c.RALifeCycleManagerImpl.startRA(
 at com.ibm.ejs.j2c.RALifeCycleManagerImpl.resourceProviderEvent(
. . .
```

ManagedConnectionFactory beans

ManagedConnectionFactory JavaBeans are validated during the initial Java Naming and Directory Interface (JNDI) lookup of a J2C connection factory.

When validation fails, the Application Server rejects the ManagedConnectionFactory instance and displays a naming exception to the application that performs the lookup. This exception indicates the causal constraint violation exception (javax.validation.ConstraintValidationException).

Applications cannot establish outbound connections to the resource. In-doubt transactions started over connections to the resource that were created by the connection factory cannot be recovered.

The following example is a ManagedConnectionFactory bean, MyMcf, at heap address 7dd07dd0, Two validation constraints are violated. The constraint violation exception causes the application to not obtain a connection factory that is required to create a connection to the resource, MyConnector:

```
[9/30/10 7:58:58:734 CDT] 00000023 BeanValidatio E
 J2CA0238E: JavaBean com.my.company.adapter.MyMcf@7dd07dd0 failed Bean Validation due to one or more invalid
  configuration settings indicated in the following list of constraint violations:
 ConstraintViolationImpl{interpolatedMessage='must be less than or equal to 30', propertyPath=mcfProperty2,
    rootBeanClass=class com.my.company.adapter.MyMcf, messageTemplate='{javax.validation.constraints.Max.message}'}
 ConstraintViolationImpl{interpolatedMessage='The value should be greater than 10', propertyPath=mcfProperty4,
    rootBeanClass=class com.my.company.adapter.MyMcf, messageTemplate='The value should be greater than 10'}
....
[9/30/10 7:58:58:765 CDT] 00000023 ConnectionFac E
 J2CA0009E: An exception occurred while trying to instantiate the ManagedConnectionFactory class com.my.company.adapter.MyMcf
  used by resource j2c/MyConnector : com.ibm.ejs.j2c.metadata.ConstraintViolationException
 at com.ibm.ejs.j2c.metadata.BeanValidationHelper.validate(
 at com.ibm.ejs.j2c.ServerFunction.validate(
 at com.ibm.ejs.j2c.J2CUtilityClass.createMCFEntry(
 ...
 at javax.naming.InitialContext.lookup(
 at com.my.company.app.MyEjbImpl.testJbv(
. . .
```

ActivationSpec bean violations

ActivationSpec beans are validated when the applications starts. This is when the Application Server initially activates message endpoints bound to J2C activation specifications. These activation specifications name the bean class in their configuration. When validation fails, the endpoint fails to activate and the resulting constraint violation exception causes the application hosting the endpoint to fail.

Because the J2C resource adapter that contains the activation specification is started, applications can still establish connections to the resource. The resource can deliver messages to endpoints that have successfully activated. If the activation specification is defined within an embedded resource adapter, the server stops the adapter in the course of stopping the application. Failed transactional messages delivered by previous instances of the resource adapter that contains the activation specification cannot be recovered.

The following example is an ActivationSpec bean, MyActSpec, at heap address 51625162. Two validation constraints are violated. The log shows the constraint violation exception that causes the application, my_company_app, to fail:

```
[9/29/10 10:52:05:125 CDT] 00000009 BeanValidatio E
 J2CA0238E: JavaBean com.my.company.adapter.MyActSpec@51625162 failed Bean Validation due to one or more invalid
  configuration settings indicated in the following list of constraint violations:
 ConstraintViolationImpl{interpolatedMessage='Size should be between 2 and 4', propertyPath=asProperty1,
    rootBeanClass=class com.my.company.adapter.MyActSpec, messageTemplate='Size should be between 2 and 4'}
 ConstraintViolationImpl{interpolatedMessage='Should be < 30', propertyPath=asProperty2,
    rootBeanClass=class com.my.company.adapter.MyActSpec, messageTemplate='Should be < 30'}
[9/29/10 10:52:05:171 CDT] 00000009 RAWrapperImpl E
 J2CA0089E: The method activateEndpoint on ResourceAdapter JavaBean
   cells/IBM-46DF84D297BNode01Cell/nodes/IBM-46DF84D297BNode01/resources.xml#J2CResourceAdapter_1285109389828
 failed with the following exception:
  javax.resource.ResourceException: com.ibm.ejs.j2c.metadata.ConstraintViolationException
 at com.ibm.ejs.j2c.ActivationSpecWrapperImpl.validateActivation...( at com.ibm.ejs.j2c.ActivationSpecWrapperImpl.createAndInitializ...(
 at com.ibm.ejs.j2c.ActivationSpecWrapperImpl.activateEndpoint(
...
[9/29/10 10:52:05:750 CDT] 00000009 ApplicationMg A   WSVR0217I: Stopping application: my_company_app
. . .
```

AdministeredObject beans

AdministeredObject beans are validated when the server starts a J2C resource adapter that contains the administered object in its configuration. When validation fails, the server rejects the AdministeredObject instance and the resulting constraint violation exception causes the resource adapter to fail.

The following example is an AdministeredObject beans, MyAdminObj, at heap address 3a803a80. Two validation constraints are violated. The log shows the constraint violation exception that causes resource adapter to fail:

```
[9/29/10 10:51:25:125 CDT] 00000000 BeanValidatio E
 J2CA0238E: JavaBean com.my.company.adapter.MyAdminObj@3a803a80 failed Bean Validation due to one or more invalid
  configuration settings indicated in the following list of constraint violations:
 ConstraintViolationImpl{interpolatedMessage='The value should be greater than 10', propertyPath=aoProperty4,
  rootBeanClass=class com.my.company.adapter.MyAdminObj, messageTemplate='The value should be greater than 10'}
...
[9/29/10 10:51:25:218 CDT] 00000000 AdminObjectSe A
 J2CA0017I: An exception occurred while building the serializable for JNDI deployment of jms/MyAdminObj :
 com.ibm.ejs.j2c.metadata.ConstraintViolationException
 at com.ibm.ejs.j2c.metadata.BeanValidationHelper.validate(
 at com.ibm.ejs.j2c.metadata.BeanValidationHelper.validate(
 at com.ibm.ejs.j2c.AdminObjectSerBuilderImpl._createAndValidate...(
 at com.ibm.ejs.j2c.AdminObjectSerBuilderImpl.createAndValidate...(
 at com.ibm.ejs.j2c.RALifeCycleManagerImpl.startRA(
. . .
```

JCA 1.6 support for annotations in RAR modules:

Note: The Java Connector Architecture (JCA) Version 1.6 specification adds support for Java annotations in resource archive (RAR) modules. Annotations are a means of specifying metadata, or configuration data, for a RAR module in the class files that make up the module.

Before JCA 1.6, this metadata was specified only in the deployment descriptor, but now you can specify this metadata using either a deployment descriptor or annotations. Metadata that is specified in annotations is merged into the deployment descriptor of a RAR module when it is updated, if the module is not marked metadata-complete in the deployment descriptor and if the module version is 1.6 or later.

The metadata-complete element defines whether the deployment descriptor for the resource adapter module is complete or whether the class files that are available to the module and packaged with the resource adapter should be examined for annotations that specify deployment information. If the metadata-complete is set to *true*, the application server deployment tool must ignore any annotations that specify deployment information, which might be present in the class files of the application. If metadata-complete is not specified, or is set to *false*, the deployment tool must examine the class files of the application for annotations, as specified by the JCA 1.6 Specification. If the deployment descriptor is not included, or is included but not marked metadata-complete, the deployment tool processes annotations.

Application servers must assume that metadata-complete is *true* for resource adapter modules with deployment descriptors that meet the requirements of JCA specification 1.5 and earlier. For a complete list of the supported annotations and their usage, consult the JCA specification.

Note: The JCA Version 1.6 specification also adds support for Bean Validation constraint annotations in RAR modules. You can specify Bean Validation constraint metadata for RAR JavaBeans by decorating your classes with Bean Validation constraint annotations or by supplying XML validation descriptors. The Application Server validates the constraints of all JCA 1.6 RAR JavaBeans instances before placing them into service at run time.

Connection factory JNDI name practices

Observe the conventions of the Java Naming and Directory Interface (JNDI) service in WebSphere Application Server when you create connection factory JNDI names.

Distributed computing environments often employ naming and directory services to obtain shared components and resources. Naming and directory services use name-to-object mappings to associate names with objects such locations, services, information, and resources. The Java Naming and Directory Interface (JNDI) provides a common interface that is used to access the various naming and directory services.

Naming your resources indirectly

When creating a connection factory or data source, a JNDI name is given by which the connection factory or data source can be looked up by a component. WebSphere Application Server uses an *indirect* name with the `java:comp/env` prefix:

- When you create a WebSphere Application Server data source, the default JNDI name is set to `jdbc/data_source_name`.
- When you create a connection factory, its default name is `eis/j2c_connection_factory_name`.

If you override these values by specifying your own, retain the `java:comp/env` prefix. An indirect name makes any resource-reference data associated with the application available to the connection management runtime, to better manage resources based on the res-auth, res-isolation-level, res-sharing-scope, and res-resolution-control settings.

Naming your resources for use with CMP

In addition, if you click the checkbox for the **Use this data source for container managed persistence (CMP)** option when you create the data source, another reference is created with the name of `eis/jndi_name_of_datasource_CMP`. For example, if a data source has a JNDI name of *jdbc/myDatasource*, the CMP JNDI name is *eis/jdbc/myDatasource_CMP*. This name is used internally by CMP and is provided simply for informational purposes.

Establishing custom finder SQL dynamic enhancement server-wide

Enable support for dynamic SQL enhancement of all custom finders, defined in all beans, by modifying the custom properties of your application server in the administrative console.

About this task

To establish this support on a server-wide basis (that is, dynamic SQL enhancement of all custom finders defined in all beans is enabled), use the following steps.

Procedure

1. Open the administrative console.
2. Select **Servers**.
3. Select **Application Servers**.
4. Select the server you want to configure.
5. In the Additional Properties area, select **Process Definition**.
6. In the Additional Properties area, select **Control** or **Servant**. Select **Control** to define the property in the Control, **Servant** to define the property in the Servant.
7. In the Additional Properties area, select **Java Virtual Machine**.
8. Select **Custom Properties**.
9. Select **com.ibm.websphere.ejbcontainer.customfinder.honorAccessIntent** and enter a value of **all**. If the property is not present in the list, create a new property name, enter the name *com.ibm.websphere.ejbcontainer.customfinder.honorAccessIntent* and the value **all**.

Establishing custom finder SQL dynamic enhancement on a set of beans

You can enable support for all custom finders defined on beans by modifying your application server's custom properties through the administrative console.

About this task

To establish this support for all custom finders defined on a set of beans use the following steps.

Procedure

1. Open the administrative console.
2. Select **Servers**.
3. Select **Application Servers**.
4. Select the server you want to configure.
5. Select **Server Infrastructure** > **Java and Process Management** > **Process Definition**.
6. Select **Java Virtual Machine**.
7. Select **Custom Properties**.
8. Select **com.ibm.websphere.ejbcontainer.customfinder.honorAccessIntent** and enter a value that corresponds to a list of beans that need this support, with each bean's name separated from the others by a colon (:). For example, *beanA:beanB:beanC*.

   If the property is not present in the list, create a new property name, enter the name *com.ibm.websphere.ejbcontainer.customfinder.honorAccessIntent* and enter the list as the value.

CMP connection factories collection

Use this page to view existing container managed persistence (CMP) connection factories settings.

These connection factories are used by a CMP bean to access any backend data store. A CMP connection factory is used by EJB model 2.x Entities with CMP version 2.x. Connection factories listed on this page are created automatically under the WebSphere Relational Resource Adapter when you check the box **Use this Data Source in container managed persistence (CMP)** in the General Properties area on the Data Source page. You cannot modify the settings for a CMP connection factory, and you cannot delete CMP connection factories from this collection. To remove the CMP connection factory object, you must navigate to the data source associated with the CMP connection factory and uncheck the **Use this Data Source for CMP** check box.

To view this administrative console page:

1. Click **Resources** > **Resource Adapters** > **Resource adapters**.
2. View the built-in resources. Click **Preferences**, select **Show built-in resources** and click **Apply**.
3. Click **WebSphere Relational Resource Adapter** > **CMP connection factories**

Name

Specifies the display name for the resource.

Data type String

JNDI Name

Specifies the JNDI name of the resource.

Data type String

Description

Specifies a description for the resource.

Data type String

Category
Specifies a category string which can be used to classify or group the resource.

Data type String

CMP connection factory settings
Use this page to view the settings of a connection factory that is used by a container-managed persistence (CMP) bean to access any database server. Because the connection factory is created and managed automatically, the settings of the connection factory cannot be modified.

To view this administrative console page:

1. Click **Resources** > **Resource Adapters** > **Resource adapters**.
2. Click **Preferences**, select **Show built-in resources** and click **Apply**.
3. Click **WebSphere Relational Resource Adapter** > **CMP Connection Factories** > ***connection_factory***.

Name:

Specifies the display name for the resource.

JNDI name:

Specifies the JNDI name of the resource.

Description:

Specifies a description for the resource.

Category:

Specifies a category string which can be used to classify or group the resource.

Authentication preference:

Specifies which of the authentication mechanisms that are defined for the corresponding resource adapter applies to this connection factory. This property is deprecated starting with version 6.0.

For example, if two authentication mechanism entries are defined for a resource adapter (*KerbV5* and *Basic Password*), this specifies one of those two types. If the authentication mechanism preference specified is not an authentication mechanism available on the corresponding resource adapter, it is ignored.

Data type String

Component-managed authentication alias:

References authentication data for component-managed signon to the resource.

Data type Drop-down list

Container-managed authentication alias:

References authentication data for container-managed signon to the resource.

Data type	Drop-down list

Configuring resource adapters

You can view a list of installed and configured resource adapters in the administrative console. Also, you can use the administrative console to install new resource adapters, create additional configurations of installed resource adapters, or delete resource adapter configurations.

Before you begin

A resource adapter is an implementation of the Java EE Connector Architecture (JCA) specification. The JCA specification provides access for applications to resources outside of the server or provides access for an enterprise information system (EIS) to applications on the server. It can provide application access to resources such as DB2, Customer Information Control System (CICS), Information Management Systems (IMS™), SAP, and PeopleSoft.

It can provide an EIS with the ability to communicate with message-driven beans that are configured on the server. Some resource adapters are provided by IBM; however, third-party vendors can provide their own resource adapters. A resource adapter implementation is provided in a resource adapter archive (RAR)file; this file has an extension,`RAR`. A resource adapter can be provided as a stand-alone adapter or as part of an application, in which case it is called an embedded adapter.

The Java Connector Architecture (JCA) Version 1.6 specification adds support for Java annotations and Bean Validation in RAR modules. For more information about annotation support and metadata, see the topic, JCA 1.6 support for annotations in RAR modules.

About this task

Use this task to configure a stand-alone resource adapter archive file. Embedded adapters are installed as part of the application installation. This panel can be used to work with either type adapter.

Procedure

1. Open the product administrative console.
2. Select **Resources** > **Resource adapters** > *resource_adapter*.
3. Set the scope setting. This field specifies the level to which this resource definition is visible. For general information, see the topic, Administrative console scope settings, in the Related Reference section. The Scope field is a read-only string field that shows where the particular definition for a resource adapter is located. This field is set either when the resource adapter is installed, which can only be at the node level, or when a new resource adapter definition is added.
4. Configure the description. This field specifies a text description of the resource adapter. Use a free-form text string to describe the resource adapter and its purpose.
5. Set the archive path. Use this field to specify the path to the RAR file containing the module for this resource adapter. This property is required.
6. Set the class path. The list of paths or JAR file names that together form the location for the resource adapter classes is set here. This includes any additional libraries needed by the resource adapter. The resource adapter code base is automatically added to the class path, but if anything outside the RAR is needed it can be specified here.
7. Set the native path. The list of paths that form the location for the resource adapter native libraries is set here. The resource adapter code base is automatically added to the class path, but if anything outside the RAR is needed it can be specified here.
8. Set the ThreadPool alias. The name of a thread pool that is configured in the server that is used by the resource adapter Work Manager is specified in this field. If there is no thread pool configured in the

server with this name, the default configured thread pool instance, named Default, is used. This property is only necessary if this resource adapter uses Work Manager. This field does not apply for the z/OS platform.

Resource adapters collection

Use this panel to perform the following actions on stand-alone resource adapters: view the list of installed resource adapters, install additional resource adapters, create additional configurations of already installed resource adapters and delete resource adapter configurations.

A resource adapter can be provided as a stand-alone adapter or as part of an application, in which case the resource adapter is referred to as an embedded adapter. Refer to related task, Installing resource adapters within applications, for more information on embedded resource adapters. A resource adapter is an implementation of the Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) specification. Enterprise applications can use a resource adapter to access resources outside of the application server including relational databases like DB2, online transaction processing (OLTP) systems like CICS, and enterprise information system (EIS) like SAP and PeopleSoft. A resource adapter can provide an EIS with the ability to communicate with message-driven beans (MDB) that are configured on the server. Resource adapters are provided by IBM or third party vendors. A resource adapter implementation is provided in a resource adapter archive file; this file has an extension of `.rar`.

To view this administrative console page, click **Resources** > **Resource Adapters** > **Resource adapters**.

To display a list of all of the resource adapters that are defined for a specific scope, select that scope.

To view the stand-alone resource adapters that are provided with the application server, select the **Show built-in resources** checkbox in the **Preferences** section.

To view additional information about, or to change the settings of a specific resource adapter, click the resource adapter name.

To perform an action on a specific resource adapter, select the checkbox beside the resource adapter name and click the appropriate button detailed below.

Install RAR:

Install a resource archive (RAR).

You can upload a RAR file from the local file system, or specify a RAR file on a remote file system. The RAR file must be installed at the node level.

New:

Create a copy of the selected resource archive which is already installed on the application server.

If you want to create a copy of an installed resource adapter, specify a server for the scope, and click **New**. You cannot create a copy of a resource adapter at the node scope. If you want to install a new resource adapter, click **Install RAR**.

Delete:

Delete the selected resource adapter.

Update RAR:

Update the selected resource adapter. Update a resource adapter archive (RAR) file when you determine that a resource adapter, or a set of resource adapters, needs to be updated with a different version or implementation.

Different versions or implementations of resource adapters can include different settings, therefore, updating your adapter might be beneficial if you require a specific set of configuration options. You can update the resource adapter for all of the nodes in a cell or all the nodes in a cluster. If some of your nodes are earlier than Version 7.0, the RAR update is not supported until those nodes are migrated to Version 7.0 or later.

Name:

Specifies the name of the resource adapter.

Description:

Specifies a text description of the resource adapter.

This description is a free-form text string to describe the resource adapter and its purpose.

Scope:

Specifies the level at which this resource adapter is visible. For general information, read about administrative console scope settings.

Some considerations that you should keep in mind for this particular panel are:

- Changing the scope enables you to see which resource adapter definitions exist at that level.
- Changing the scope does not have any effect on installation. Installations are always done under a scope of node, no matter what you set the scope to.
- When you create a new resource adapter from this panel, you must change the scope to what you want it to be before you click **New**.

Resource adapter settings:

Use this page to specify settings for a resource adapter.

A resource adapter is an implementation of the Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) specification that provides access for applications to resources outside of the server, provides access for applications to an enterprise information system (EIS), or provides access for an EIS to applications on the server. Resource adapters provide applications access to resources such as DB2, CICS, SAP and PeopleSoft. Resource adapters can provide an EIS with the ability to communicate with message driven beans that are configured on the server. Some resource adapters are provided by IBM; however, third party vendors can provide their own resource adapters. A resource adapter implementation is provided in a resource adapter archive file (RAR); this file has an extension of `.rar`. A resource adapter can be provided as a stand alone adapter or as part of an application, in which case the resource adapter is referred to as an embedded adapter.

The JCA Version 1.6 specification adds support for Java annotations in RAR modules. For more information on annotation support see the topic, JCA 1.6 support for annotations in RAR modules.

To view this administrative console page, click one of the following paths:

- **Resources** > **Resource Adapters** > **Resource adapters** > **New**.
- **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter***.
- **Applications** > **WebSphere enterprise applications** > ***enterprise_application*** > **Manage Modules** > ***connector_module*** > **Resource Adapter**.

- Install a new resource adapter archive:
  1. Click **Resources** > **Resource Adapters** > **Resource adapters** > **Install RAR**.
  2. Specify a full path for the local file system or remote file system, and click **Next**.

Scope:

Specifies the highest topological level at which application servers can use this adapter.

The Scope field is a read-only string field that specifies where the particular definition for a resource adapter is located. The Scope field is set when the resource adapter is installed, which can only be at the node level, or when a new resource adapter definition is added.

Name:

Specifies the name of the resource adapter definition.

This property is a required string containing no spaces that is a meaningful text identifier for the resource adapter.

Description:

Specifies a text description of the resource adapter.

This description is a free-form text string to describe the resource adapter and its purpose.

Archive path:

Specifies the path to the installed resource archive file that contains the module for this resource adapter.

You can only select RAR files that are installed on the nodes within the selected scope, preventing you from configuring a selection that might fail for some of your nodes.

Note: For resources at the cell scope, the RAR files that are available are those that are installed on each individual node in the entire cell. For resources at a cluster scope, the RAR files that are available are those that are installed on each individual node in that particular cluster.

This property is required.

Data type String

Class path:

Specifies a list of paths or Java archive file (JAR) names that together form the location for the resource adapter classes.

Class path entries are separated by using the ENTER key and must not contain path separator characters like ';' or ':'. Class paths can contain variable (symbolic) names that can be substituted using a variable map. Check your driver installation notes for specific JAR file names that are required.

Native library path:

Specifies an optional path to any native libraries, which are .dll or .so files.

Native path entries are separated by using the ENTER key and must not contain path separator characters like ';' or ':'. Native paths can contain variable (symbolic) names that can be substituted using a variable map.

Isolate this resource provider:

Specifies that this resource provider will be loaded in its own class loader. This allows different versions of the same resource provider to be loaded in the same Java Virtual Machine. Give each version of the resource provider a unique class path that is appropriate for that version.

Ensure that all copies of a resource adapter have the same value for this option. For example, if you create a resource adapter at the cluster scope, the value of this option will be taken from the resource adapter archive (RAR) that you copy. When you create the copy, you cannot modify the value for any instances of that RAR, which would be the copies at the node or cluster scope in this example. If you need to modify the value, you have to delete the copies of the RAR until there is only one instance of that particular RAR that is left.

Note: You cannot isolate a resource provider if you specify a native library path.

Thread pool alias:

Specifies the name of a thread pool that is part of the server configuration for this resource adapter. Set this property only if the resource adapter uses the work manager service.

If you input a thread pool name that does not exist in the server configuration, the application server uses the name DEFAULT.

Advanced resource adapter properties:

Use this page to specify advanced settings for resource adapters that comply with the Version 1.5 and 1.6 Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) specification.

A resource adapter is an implementation of the JCA specification that provides access for applications to an enterprise information system (EIS), like DB2, CICS, SAP and PeopleSoft, or provides access for an EIS to applications on the server. A resource adapter can also provide an EIS with the ability to communicate with message-driven beans that are configured on the server. Some resource adapters are provided by IBM, but third party vendors can provide their own resource adapters.

A resource adapter implementation is provided in a resource adapter archive file; this file has an extension of .rar. A resource adapter can be provided as a stand-alone adapter or as part of an application, in which case it is referred to as an embedded adapter.

To view this administrative console page, click **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **Advanced resource adapter properties**.

Restrict the JVM to allow only one instance of this resource adapter:

Prevents more than one instance of a resource adapter JavaBeans with a unique resource adapter implementation class name from existing in the same Java Virtual Machine (JVM). This field is only available on resource archives that allow definitions for activation specifications.

Note: Enabling this setting imposes a restrictive condition on the inbound communications. For example, if two applications embed the same resource adapter, only the first application to start will be able to access resources through its embedded resource adapter. If a stand-alone resource adapter is configured for a single instance, no applications that embed that same resource adapter will be able to access resources.

Data type	Boolean (checkbox)
Default	False (disabled)

Register this resource adapter with the high availability manager:

Specifies that the high availability (HA) manager will manage the lifecycle of a JCA resource adapter in a cluster. This option is only applicable to resource adapters with a version greater than JCA 1.0 and running on the Network Deployment version of WebSphere. Do not select this option without first consulting the product documentation for the resource adapter, because this option requires the resource adapter to support high availability of inbound messaging. This field is only available on resource archives that allow definitions for activation specifications.

Note: Enabling this setting imposes a restrictive condition on the inbound communications.

This setting can be implemented with:

- **Endpoint failover:** allows only one resource adapter in an HA group to receive messages across multiple servers. The result is that only one resource adapter can have endpoints active at one time.
- **Resource adapter instance failover:** allows only one resource adapter in an HA group to be started across multiple servers. Inbound or outbound communication is limited to one resource adapter in the cluster.

Data type	Boolean (checkbox with implementation options)
Default	False (disabled)

Directory conventions:

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
: The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
: The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
: The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
: The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 19. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root
The default Web Server Plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V8/webserver directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, was.install.library, and is located under the *app_server_root* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the /QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/*profile_name* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server - Express is the /QIBM/UserData/WebSphere/AppServer/V8/Express directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Updating a stand-alone resource adapter archive

Use the resource adapter archive (RAR) update wizard to update the stand-alone RAR files to a newer version. The application server uses the classes and other code that comprise a resource adapter archive to support the resource adapters that you configure.

Before you begin

A resource adapter must be installed on the application server, and you must have a new version of the resource adapter that is compatible with the old version. You can create the RAR file with an assembly tool, or the vendor for the resource adapter can provide the new version. If the new version of a RAR file is not compatible with the old version of the resource adapter, an update is not possible.

The Java Connector Architecture (JCA) Version 1.6 specification adds support for Java annotations in RAR modules. For more information on annotation support and metadata, see the topic JCA 1.6 support for annotations in RAR modules.

About this task

Update a RAR file when you determine that a resource adapter or a set of resource adapters needs to be updated with a different version. Different versions of resource adapters can include different settings, so updating your adapter might be beneficial if you require a certain set of configuration options. It is your choice to update the resource adapter for all the nodes in a cell or all the nodes in a cluster. If some of your nodes are earlier than Version 7.0, the RAR update is not supported until those nodes are migrated to Version 7.0. Updates to JCA 1.6 adapters are only applicable to a Version 8.0 node.

If you prefer to have more than one version of a resource adapter active in a given Java Virtual Machine (JVM), the update wizard does not provide the option of creating another version and keeping the old. In this case, you need to create an isolated resource adapter and configure it accordingly. Refer to the topic on configuring a resource adapter for more information.

Procedure

1. Save all configuration changes.
2. Back up your configuration settings with the backupConfig tool. The backupConfig tool is located in the `app_server_root/bin` directory. Read the topic on the backupConfig command for more information about how to use this command.
3. Stop any servers that contain the RAR file that is updated. Refer to the topic, Administering applications and their environment, for information about how to stop or start an application server.
4. Click **Resources** > **Resource Adapters** > **Resource Adapters**.
5. Select the check box next to the RAR file to update, and click **Update RAR**.
6. Specify the installation path for the RAR file, and click **Next**.
   - If your RAR file is located on the same workstation as your browser, select **Local file system**, and browse to find the file.
   - If your RAR file is located on the server workstation where the application server is installed, select **Remote file system**, and specify the fully qualified path to the file.
7. Review the configuration information that is provided for the RAR file. The following information is displayed for the RAR file:
   - Name
   - Current® RAR version
   - New RAR version
   - Scope
   - Any existing copies of the resource adapter. The resource adapters with an asterisk (*) are copies of the resource adapter and must also be updated at the same time.

   Click **Next** when you are finished.
8. Review the summary panel, and click **Finish** when you are satisfied with the configuration settings. When you click Finish, all the configuration changes are saved automatically. To revert to an older version of the resource adapter you must perform the update process again, and specify the older version of the RAR file.
9. Restart the servers that contain the updated RAR file.

What to do next

If you are not satisfied with the results, and you backed up your configuration with backupConfig tool, use the restoreConfig tool to restore your backup configuration. Read the topic on the restoreConfig command

for more information about how to use this command.

RARUpdate command group

Use the resource adapter archive (RAR) update wizard to upgrade the RAR module and property configuration of J2C Resource Adapters. The application server uses the classes and other code that comprise a resource adapter archive to support the resource adapters that you configure.

To avoid problems, use the administrative console RAR update wizard. For more information, see the topic Updating a resource adapter archive.

The RARUpdate command group contains the following commands:

- "compareResourceAdapterToRAR"
- "getNewRAObjectProperties"
- "findOtherRAsToUpdate" on page 246
- "updateRAR" on page 246

compareResourceAdapterToRAR

The compareResourceAdapterToRAR command determines whether a resource adapter is compatible with a new RAR file. A resource adapter may be updated with a RAR only when the two are compatible.

Target object

Resource adapter Object ID - The configuration object of the resource adapter that will be compared for update.

Result

The command returns true if the resource adapter is compatible with the specified RAR. It also displays the versions of the resource adapter and new RAR file. If the resource adapter is not compatible with the specified RAR, then the command returns false and provides a message explaining why they are not compatible.

Required parameters

-rarPath
The absolute path to a RAR file. (String, required)

Examples

Batch mode example usage:

```
AdminTask.compareResourceAdapterToRAR("Test Resource Adapter(cells/cell/nodes/node|resources.xml#J2CResourceAdapter_1169157308943)",
     '[-rarPath "c:\tra\rar\TRA.rar"]')
```

getNewRAObjectProperties

The getNewRAObjectProperties command obtains the list of new properties within a RAR file that may be configured when updating a resource adapter.

Target object

Resource adapter Object ID - The configuration object of the resource adapter that will be updated.

Result

You receive the list of new properties that may be configured on the resource adapter when updating a RAR.

Required parameters

-rarPath
The absolute path to a RAR file. (String, required)

-returnType
The type of value to return, "String" or "Hashtable". When returnType is "String", the command returns a String inteded for input to the updateRAR command; if returnType is "Hashtable", then the command returns a java.util.Hashtable which is intended for use with the Admin console. The returnType defaults to "String". (String, optional)

Examples

Batch mode example usage:

- Using Jython:

```
AdminTask.getNewRAObjectProperties('"Test Resource Adapter(cells/cell/nodes/node|resources.xml#J2CResourceAdapter_1169157308943)"',
     '[-rarPath c:/tra/RAR/TRA.rar -returnType String]')
```

findOtherRAsToUpdate

The findOtherRAsToUpdate command locates other resource adapters in the configuration that are similar to the resource adapter to be updated. The resulting resource adapters should also be updated.

Target object

Resource adapter Object ID - The configuration object of the resource adapter that will be compared for update.

Result

The command returns a list of resource adapter Object ID Strings that may be input as the target object of RARUpdate commands.

Examples

Batch mode example usage:

- Using Jython:

```
AdminTask.findOtherRAsToUpdate('"Test Resource Adapter(cells/cell/nodes/node|resources.xml#J2CResourceAdapter_1169157308943)"')
```

updateRAR

The updateRAR command updates the RAR and configuration of a resource adapter at a specific scope.

Target object

Resource adapter Object ID - The configuration object of the resource adapter that will be updated.

Result

This command returns a message of success or failure upon completion.

Required parameters

-rarPath
The absolute path to the new RAR file. (String, required)

Optional parameters

-ResourceAdapterProps
A list of [name value] pairs of new properties to set on the ResourceAdapter implementation class. You can specify the following parameters for this step:

-name The name of the resource adapter property. (String, optional)

-value The value of the resource adapter property. (String, optional)

-ConnectionFactoryProps
A list of [id name value] triplets of new properties to set on the connection factories in the resource adapter configuration. You can specify the following parameters for this step:

-id The Java Naming and Directory Interface (JNDI) name of the connection factory. (String, optional)

-name The name of the connection factory property. (String, optional)

-value The value of the resource adapter property. (String, optional)

-ActivationSpecProps
A list of [id name value] triplets of new properties to set on the activation specifications in the resource adapter configuration. You can specify the following parameters for this step:

-id The Java Naming and Directory Interface (JNDI) name of the activation specification. (String, optional)

-name The name of the activation specification property. (String, optional)

-value The value of the activation specification property. (String, optional)

-AdminObjectProps
A list of [id name value] triplets of new properties to set on the administered objects in the resource adapter configuration. You can specify the following parameters for this step:

-id The Java Naming and Directory Interface (JNDI) name of the activation specification. (String, optional)

-name The name of the administered object property. (String, optional)

-value The value of the administered object property. (String, optional)

Examples

Batch mode example usage:

- Using Jython:

```
AdminTask.updateRAR('"Test Resource Adapter(cells/cell/nodes/node|resources.xml#J2CResourceAdapter_1169157308943)"',
     '[-rarPath c:/tra/RAR/TRA.rar -ResourceAdapterProps [[RAProp1 RAProp1Val] [RAProp2 RAProp2Val]]
     -ConnectionFactoryProps [[eis/TRA_CF_1 CFProp1 CFProp1Val] [eis/TRA_CF_2 CFProp1 CFProp1Val2] [eis/TRA_DIF_CF CFProp2 CFProp2val]]
     -ActivationSpecProps [[eis/TRA_AS_1 ASProp1 ASProp1Val] [eis/TRA_AS_1 ASProp2 ASProp2Val] [eis/TRA_AS_2 ASProp3 ASProp3val] [eis/TRA_AS_2 ASProp4 ASProp4Val
     -AdminObjectProps [[eis/TRA_AO_1 AOProp1 AOProp1Val] [eis/TRA_AO_1 AOProp2 AOProp2Val] [eis/TRA_AO_2 AOProp1 AOProp1Val] [eis/TRA_AO_3 AOProp2 AOProp2Val]]]
```

Interactive mode example usage:

- Using Jython:

```
AdminTask.updateRAR('-interactive')
```

Mapping resource manager connection factory references to resource factories

You can use the administrative console to bind the resource manager connection factory references to one of the configured resource factories.

Before you begin

Before you can map the resource manager connection factory references to a configured resource factory, your enterprise application must contain configured resource references. You must use an assembly tool, such as Rational Application Developer, to assemble the application before deploying it through the administrative console.

Important: If your application does not contain resource references, the Resource references link does not display in the administrative console.

For more information on resource references, see the following topics:

- Creating or changing a resource reference
- Resource reference benefits

About this task

If the value of the res-auth element is `Container` within the deployment descriptor for your application, then you must specify the mapping configuration.

Procedure

1. Click **Applications** > **Enterprise applications >** *application_name*.
2. From Resources, select **Resource references**.
3. Select the application module and specify an authentication method for the selected connection factory reference binding. Select either **Use default method**, **Use custom login configuration** or **Use trusted connections**. If you select the **Use default method** option, the DefaultPrincipalMapping login configuration is selected. If you select the **Use trusted connections** option, then the TrustedConnectionMapping login configuration is selected. You must select an authentication data alias from the list.
4. After you make a selection, click **Apply** for the configuration to take effect.
5. If you select the **Use trusted connection option**, then you must select an authentication data alias from the menu list. The alias that is specified is what the application server uses to get the initial trusted connection.
6. Click **Apply**. The selected login configuration name and an **Mapping properties** button is displayed in the Login configuration field of the particular connection factory reference binding.
7. Click **Mapping properties > New** to specify the properties for your configuration. Click **OK** after specifying the properties on the Mapping properties panel.
8. If you select the **Use trusted connection** option, then you must select an authentication data alias from the menu list.
9. Click **Apply**. The **Mapping properties** button is displayed in the login configuration field of the particular connection factory reference binding.
10. Click **Mapping properties** to modify the properties of the trusted connection. See the topic, Setting the security properties for trusted connections, for information on tuning the mapping properties for the trusted connection.
11. Click **OK** and **Save** on the Resource references panel to save your changes to the master configuration.

Managing messages with message endpoints

Manage message delivery for message-driven beans (MDB) that are deployed as message endpoints. The message endpoints are managed beans (MBeans) for inbound resource adapters that are compliant with Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) Version 1.5.

About this task

The application server provides message endpoint MBeans to assist you in managing the delivery of a message to your message-driven beans that are acting as listeners on specific endpoints, which are destinations, and in managing the enterprise information system (EIS) resources that are utilized by these message-driven beans. Message-driven beans that are deployed as message endpoints are not the same as message-driven beans that are configured against a listener port. Message-driven beans that are used as message endpoints must be deployed using an ActivationSpecification that is defined within a resource adapter configuration for JCA Version 1.5.

With message endpoint MBeans, you can activate and deactivate specific endpoints within your applications to ensure that messages are delivered only to listening message-driven beans that are interacting with healthy EIS resources. This capability allows you to optimize the performance of your JMS applications in situations where an EIS resource is not behaving as expected. Message delivery to an endpoint typically fails when the message driven bean that is listening invokes an operation against a resource that is not healthy. For example, a messaging provider, which is an inbound resource adapter that is JCA Version 1.5 compliant, might fail to deliver messages to an endpoint when its underlying message-driven bean attempts to commit transactions against a database server that is not responding.

Note: Design your message-driven beans to delegate business processing to other enterprise beans. Do not access the EIS resources directly in the message-driven bean, but do so indirectly through a delegate bean.

Message endpoint MBeans alleviate two problems that are inherent to applications that provide message endpoints that access resources:

- Failed messages require additional processing, such as delivering them to the listening endpoint again or redirecting them to alternate destinations that process failed messages. In addition, a resource adapter might redeliver a message to an endpoint an infinite number of times.
- Message redirection requires the implementation of specialized destinations (queues and listeners) to process failed messages, as well as the logic to detect message failures. Message redirection is potentially error prone and computationally expensive due to its complexity.

The capability to deactivate (pause) and reactivate (resume) a specific message endpoint alleviates these problems by enabling the administrator to deactivate the endpoint from processing messages that are destined to fail. When the message endpoint is deactivated, you can repair the resource that is causing the problems and reactivate the endpoint to resume handling message requests. In the course of troubleshooting, you will not affect the resource adapter or the application that is hosting the endpoint.

If you are connecting to WebSphere MQ, you can also use the WAS_EndpointInitialState custom property in the activation specification to make the message endpoint start out in a deactivated state. When you set this property to Inactive, the message-driven bean connects with the destination, but does not start receiving messages. Use this setting to automatically deactivate a message endpoint when you know that certain tasks must be completed, services must be started, or checks must be carried out, before message handling begins. You activate the message endpoint in the same way as you would reactivate a message endpoint that you paused during its operation.

Procedure

1. Using the administrative console, navigate to the Message Endpoints panel for the application that is hosting the message endpoint.
   a. Select the **Applications** > **Application Types** > **Websphere enterprise applications** > ***application_name***.
   b. Select the **Runtime** panel.
   c. Select **Message Endpoints**. The panel lists the set of message endpoints that are hosted by the application.

2. Optional: Temporarily disable a message endpoint from handling messages and troubleshoot the problem.
   a. Deactivate the message endpoint by selecting the appropriate endpoint and clicking **Pause**.
   b. When the message endpoint is inactive, diagnose and repair the underlying cause of the delivery failures.
   c. Reactivate the message endpoint by selecting the appropriate endpoint and clicking **Resume**.
3. Optional: Activate a message endpoint that started out in a deactivated state. Select the appropriate endpoint and click **Resume**.

Results

The behavior you will observe when you deactivate (pause) a message endpoint using the message endpoint MBean is dependent upon a variety of factors, including the resource adapter that manages the message endpoint, the configuration of the message endpoint and the application server topology. Some specific examples of interest are as follows:

- **MDB listening on a non-durable topic (dependent on configuration):** The behavior that is implied by the deactivation (pause) of a message endpoint is often dependent upon the function that it is fulfilling. For example, if you have configured a message-driven bean to listen on a non-durable topic on the service integration bus, deactivating the message endpoint is analogous to stopping the application and will cause the subscription to be closed. This means that any messages that are published during the time that the message endpoint is paused will not be received by the message-driven bean.
- **Clustered message-driven bean (dependent on topology):** In this scenario a message-driven bean application has been deployed to a cluster of servers. A given message endpoint MBean controls only the behavior of the MDB in one server from the cluster, so will cause only one server to stop processing messages. Depending upon the messaging configuration and the specific resource adapter in use the messages that would have been consumed by the paused message endpoint may be consumed by the active message endpoints in the cluster, or they may remain unconsumed until the paused message endpoint is resumed.
- **Clustered message-driven bean, a non-clustered queue:** In this scenario, you have a cluster of servers with the same message-driven bean deployed to them. This is similar to the case, in which you have different message-driven beans with the same message selection criteria, except that in this case the message-driven beans are logically the same message-driven bean. Pausing the endpoint will cause only one of the servers to stop receiving messages, and the other message-driven beans will receive all the messages; none of the messages will be orphaned. To stop all of the endpoints, you must direct each server in the cluster to stop the local message endpoint.
- **Clustered message-driven bean, clustered queue:** In this scenario, each message-driven bean is pulling messages from a different partition of the queue. Messaging through WebSphere MQ and the Service Integration Bus have similar, but different, capabilities. If you are using WebSphere MQ, then pausing one endpoint will not allow the other instances of the message-driven bean to receive the messages. In the Service Integration Bus, messages from a paused endpoint will be redirected to the other message-driven beans.

Manage message endpoints

Use this panel to manage situations where messaging providers fail to deliver messages to their intended destinations. For example, a provider might fail to deliver messages to a message endpoint when its underlying message driven bean attempts to commit transactions against a database server that is not responding.

To view this administrative console panel:

1. Select the **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Select the **Runtime** panel. You will only see the Runtime panel if you have an application installed that is hosting message-driven beans.

3. Select **Message Endpoints**. The panel lists the set of message endpoints that are hosted by the application.

Name:

Specifies the name of the message endpoint.

Click the name of the message endpoint to view the configuration binding for the underlying endpoint message-driven bean and Activation Specification.

Running object scope:

Specifies the server where the endpoint is running.

For more information on scope, see the topic, Administrative console scope settings.

Status:

Indicates whether the message endpoint is active or inactive.

Click **Pause** to deactivate a message endpoint and stop it from handling messages.

Click **Resume** to reactivate a message endpoint that is inactive.

Configuring a JDBC provider and data source

For access to relational databases, applications use the Java Database Connectivity (JDBC) drivers and data sources that you configure for the application server.

Before you begin

Each vendor database requires different JDBC driver implementation classes for JDBC connectivity. A JDBC provider encapsulates those vendor-specific driver files. Through the data source that you associate with the JDBC provider, an application server obtains and manages the physical connections for transactions between applications and the database.

Attention: If you are accessing a DB2 database, IBM Optim pureQuery Runtime is an alternative to JDBC. For more information on pureQuery, see the topic, Task overview: IBM Optim pureQuery Runtime, in the related links section.

Before starting this task, determine the version of data source that you need according to the API specification of your applications.

- *Data sources (WebSphere Application Server Version 4)* are for use with the Enterprise JavaBeans (EJB) 1.0 specification and the Java Servlet 2.2 specification.
- Data sources of the latest standard version are for use with applications that implement the more advanced releases of these specifications.

Procedure

1. Verify that all of the necessary JDBC driver files are installed on your application server. Consult the article, Data source minimum required settings, by vendor for that information. If you opt to configure a user-defined JDBC provider, check your database documentation for information about the driver files.
2. Create a JDBC provider.

   When you create a JDBC provider from the administrative console, see the topic, Configuring a JDBC provider using the administrative console.

   OR

Using the wsadmin scripting client, see the topic, Configuring a JDBC provider using the scripting.

OR

Using the Java Management Extensions (JMX) API, see the topic, Creating a JDBC provider and data source using the JavaManagement Extensions API.

3. Create a data source.

   From the administrative console, see the topic, Creating a data source using the administrative console.

   OR

   Using the wsadmin scripting client, see the topic, Configuring new data sources using scripting. For V4 data sources, see the topic, Configuring new WAS40 data sources using scripting.

   OR

   Using the JMX API, see the topic, Creating a JDBC provider and data source using the JavaManagement Extensions API.

   Required properties: Different database vendors require different properties for implementations of their JDBC drivers. Set these properties on the WebSphere Application Server data source. Because Application Server contains templates for many vendor JDBC implementations, the administrative console surfaces the required properties and prompts you for them as you create a data source. However, if you script your data access configurations, you must consult the article Data source minimum required settings, by vendor, for the required properties and settings options.

4. Optional: Configure custom properties. Like the required properties, custom properties for specific vendor JDBC drivers must be set on the Application Server data source. Consult your database documentation for information about available custom properties. To configure a custom class to facilitate the handling of database properties that are not recognized natively by the Application Server, refer to the topic, Developing a custom DataStoreHelper class.

   You can also learn about optional data source properties in the Application Programming Guide and Reference for Java for your version of DB2 for z/OS if you use the DB2 Universal JDBC Driver provider.

5. Bind resource references to the data source. See the article, Data source lookups for enterprise beans and web modules.
6. Test the connection (for non-container-managed persistence usage). See the topic, Test connection service.

Results

If you use the DB2 JDBC Universal Driver, you might experience data source failures that the application server JVM log does not document. Check the DB2 database log or the WebSphere Application Server JDBC trace log (if JDBC trace was active). You might find that a bad authentication credential is the cause of failure. Currently the DB2 JDBC Universal Driver does not identify or surface the errors that are produced by non-valid authentication credentials in a proper or consistent way.

Even if you receive information about a bad credential, check the database and JDBC trace logs. These logs provide more reliable, detailed error data on authentication failures.

Note: The JDBC trace log exists only if the JDBC trace service is active during server start up. Activate the service in the administrative console. For more information, see the topic, Enabling trace at server startup. Specify **WAS.database** as the trace group and select **com.ibm.ws.db2.logwriter** as the trace string.

Data source minimum required settings, by vendor

These properties vary according to the database vendor requirements for Java Database Connectivity (JDBC) driver implementations. You must set the appropriate properties on every data source that you configure.

Use these tables for quick reference on the JDBC providers that represent your JDBC driver classes. Each table corresponds to a specific database vendor, product, and platform.

Following the tables are links to detailed requirements for creating data sources that correspond to each JDBC provider that the application server supports . The list includes information about connection properties that are required by the database and any optional properties that the JDBC driver supports. Use the administrative console or the wsadmin scripting tool to define these properties on your data sources.

Table 20. Apache Derby JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Apache Derby		
JDBC provider	**Transaction support**	**Version and other considerations**
Derby JDBC Provider	One-phase	• Does not support Version 4 data sources • Configurable only in nodes at version 6.0.2 and later • Not for use in clustered environment: accessible from a single JVM only
Derby JDBC Provider (XA)	One and two phase	• Does not support Version 4 data sources • Configurable only in nodes at version 6.0.2 and later • Not for use in clustered environment: accessible from a single JVM only
Derby JDBC Provider 40	One-phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources
Derby JDBC Provider 40 (XA)	One and two phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources
Derby Network Server Using Derby Client	One-phase	• Does not support Version 4.0 data sources. • Configurable only in nodes at version 6.1 and later • *Can be used in clustered environment:* a database instance can be accessed by multiple JVMs • Only for use with Apache Derby databases that run on the same node as the application server

Table 20. Apache Derby JDBC providers (continued). Use the table for quick reference on database-specific JDBC providers.

Apache Derby		
JDBC provider	**Transaction support**	**Version and other considerations**
Derby Network Server Using Derby Client (XA)	One and two phase	• Does not support Version 4 data sources • Configurable only in nodes at version 6.1 and later • *Can be used in clustered environment:* a database instance can be accessed by multiple JVMs • Only for use with Apache Derby databases that run on the same node as the application server
Derby Network Server Using Derby Client 40	One-phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources
Derby Network Server Using Derby Client 40 (XA)	One and two phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources

Table 21. DB2 on AIX, HP-UX, Linux, Solaris, and Windows systems JDBC providers. Use the table for quick reference on database-specific JDBC providers.

DB2 on AIX, HP-UX, Linux, Solaris, and Windows systems		
JDBC provider	**Transaction support**	**Version and other considerations**
DB2 Using IBM JCC Driver	One-phase	• Configurable in nodes that are at version 7.0 and later.
DB2 Using IBM JCC Driver (XA)	One and two phase	• Configurable in nodes that are at version 7.0 and later.
DB2 Universal JDBC Provider	One-phase	N/A
DB2 Universal JDBC Provider (XA)	One and two phase	N/A

Table 22. DB2 UDB for iSeries JDBC providers. Use the table for quick reference on database-specific JDBC providers.

DB2 UDB for iSeries		
JDBC provider	**Transaction support**	**Version and other considerations**
DB2 UDB for iSeries (Native)	One-phase	Recommended when you run the application server on iSeries.
DB2 UDB for iSeries (Native XA)	One and two phase	Recommended when you run the application server on iSeries.
DB2 UDB for iSeries (Toolbox)	One-phase	N/A
DB2 UDB for iSeries (Toolbox XA)	One and two phase	N/A

Table 23. DB2 on z/OS JDBC providers. Use the table for quick reference on database-specific JDBC providers.

DB2 on z/OS		
JDBC provider	**Transaction support**	**Version and other considerations**
DB2 Using IBM JCC Driver	One-phase	Configurable in version 7.0 and later nodes.
DB2 Using IBM JCC Driver (XA)	One and two phase	Configurable version 7.0 and later nodes.
DB2 Universal JDBC Provider	One-phase when connecting to the application server that is on AIX, HP-UX, Linux, Solaris, Windows, and iSeries systems	
DB2 Universal JDBC Provider (XA)	One and two phase	

Table 24. Informix JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Informix		
JDBC provider	**Transaction support**	**Version and other considerations**
Informix Using IBM JCC Driver	One phase	This provider is configurable in nodes that are at version 7.0 and later.
Informix Using IBM JCC Driver (XA)	One and two phase	This provider is configurable in nodes that are at version 7.0 and later.
Informix JDBC Driver	One-phase	N/A
Informix JDBC Driver (XA)	One and two phase	N/A
Informix using IBM DB2 JDBC Universal Driver	One phase	This provider is configurable in nodes that are at version 7.0 and later.
Informix using IBM DB2 JDBC Universal Driver (XA)	One and two phase	This provider is configurable in nodes that are at version 7.0 and later.

Table 25. Microsoft SQL Server JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Microsoft SQL Server		
JDBC provider	**Transaction support**	**Version and other considerations**
Microsoft SQL Server JDBC Driver	One-phase	N/A
Microsoft SQL Server JDBC Driver (XA)	One and two phase	N/A
DataDirect ConnectJDBC Provider type 4 driver for MS SQL Server	One-phase	N/A
DataDirect ConnectJDBC Provider, type 4 driver, for MS SQL Server (XA)	One and two phase	N/A

Table 26. Oracle JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Oracle		
JDBC provider	**Transaction support**	**Version and other considerations**
Oracle JDBC Driver	One-phase	Must use the ojdbc6.jar driver to connect to any version of Oracle database.

Table 26. Oracle JDBC providers (continued). Use the table for quick reference on database-specific JDBC providers.

Oracle		
JDBC provider	**Transaction support**	**Version and other considerations**
Oracle JDBC Driver(XA)	One and two phase	Must use the ojdbc6.jar driver to connect to any version of Oracle database.

Table 27. Sybase JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Sybase		
JDBC provider	**Transaction support**	**Version and other considerations**
Sybase JDBC 4 Driver	One-phase	jConnect v7.0
Sybase JDBC 4 Driver (XA)	One and two phase	jConnect v7.0
Sybase JDBC 3 Driver	One-phase	jConnect v6.05
Sybase JDBC 3 Driver (XA)	One and two phase	jConnect v6.05
Sybase JDBC 2 Driver	One-phase	jConnect v5.5
Sybase JDBC 2 Driver (XA)	One and two phase	jConnect v5.5

Detailed requirements

The following list identifies required class files and connection properties per JDBC provider.

After you determine the JDBC provider that suits your application and environment, ensure that you acquire the corresponding JDBC driver at a release level supported by this version of the application server. Consult the IBM support website for supported hardware and software.

Use the following links to navigate to the requirements list. Each link corresponds to a specific database vendor, product, and platform.

- Apache Derby or Cloudscape 10.x
- DB2 Universal Database for iSeries
- Informix
- Microsoft SQL Server
- Oracle
- Sybase

Data source minimum required settings for Apache Derby:

These properties vary according to the database vendor requirements for JDBC driver implementations. You must set the appropriate properties on every data source that you configure. These settings are for Apache Derby and Cloudscape data sources.

You can configure the following types of providers:

- Derby JDBC Provider
- Derby JDBC Provider (XA)
- Derby JDBC Provider 40
- Derby JDBC Provider 40 (XA)
- Derby Network Server using Derby Client
- Derby Network Server using Derby Client (XA)

- Derby Network Server using Derby Client 40
- Derby Network Server using Derby Client 40 (XA)
- **Derby JDBC Provider**

  The Derby JDBC driver provides JDBC access to the Apache Derby database by using the framework that is already embedded in the application server. You cannot use any Version 4.0 data sources with this provider.

  This provider:

  – Is configurable only in nodes at version 6.0.2 and later
  – Supports one phase data source with the following class:

    `org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource`
  – Requires the JDBC driver file:
    - derby.jar

      The full path name is ${DERBY_JDBC_DRIVER_PATH}/derby.jar. When you create a connection through the application server, the environment variables are set automatically.
  – Requires the following DataStoreHelper class:

    `com.ibm.websphere.rsadapter.DerbyDataStoreHelper`
  – Requires the following properties:

    databaseName
    The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

    If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.
- **Derby JDBC Provider (XA)**

  The Derby JDBC driver (XA) provides JDBC access to the Apache Derby database by using the framework that is already embedded in the application server.

  This provider:

  – Does not support use Version 4.0 data sources.
  – Is configurable only in nodes at version 6.0.2 and later
  – Supports the two-phase data source with the following class:

    `org.apache.derby.jdbc.EmbeddedXADataSource`
  – Requires JDBC driver file:
    - derby.jar

      The full path name is ${DERBY_JDBC_DRIVER_PATH}/derby.jar. When you create a connection through the application server, the environment variables are set automatically.
  – Requires the following DataStoreHelper class:

    `com.ibm.websphere.rsadapter.DerbyDataStoreHelper`
  – Does not require a valid authentication alias.
  – Requires the following properties:

    databaseName
    The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

    If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.

- **Derby JDBC Provider 40**

  The Derby JDBC Provider 40 provides JDBC access to the Apache Derby database by using the framework that is already embedded in the application server.

  This provider:

  – Is configurable only in nodes at version 7.0 and later.
  – Does not support Version 4.0 data sources.
  – Supports one phase data source with the following class:

    `org.apache.derby.jdbc.EmbeddedConnectionPoolDataSource40`
  – Requires the JDBC driver file:
    - derby.jar

      The full path name is ${DERBY_JDBC_DRIVER_PATH}/derby.jar. When you create a connection through the application server, the environment variables are set automatically.
  – Requires the following DataStoreHelper class:

    `com.ibm.websphere.rsadapter.DerbyDataStoreHelper`
  – Requires the following properties:

    databaseName

    The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

    If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.

- **Derby JDBC Provider 40 (XA)**

  The Derby JDBC Provider 40 (XA) provides JDBC access to the Apache Derby database by using the framework that is already embedded in the application server.

  This provider:

  – Is configurable only in nodes at version 7.0 and later.
  – Does not support Version 4.0 data sources.
  – Supports one phase data source with the following class:

    `org.apache.derby.jdbc.EmbeddedXADataSource40`
  – Requires the JDBC driver file:
    - derby.jar

      The full path name is ${DERBY_JDBC_DRIVER_PATH}/derby.jar. When you create a connection through the application server, the environment variables are set automatically.
  – Requires the following DataStoreHelper class:

    `com.ibm.websphere.rsadapter.DerbyDataStoreHelper`
  – Requires the following properties:

    databaseName

    The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

    If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.

- **Derby Network Server using Derby Client**

  Use this provider to access only Apache Derby databases that run on the same node as the application server.

This provider:

– Does not support Version 4.0 data sources.
– Is configurable only in nodes at version 6.1 and later
– Uses the following one phase data source for the Derby Network Server using Derby Client provider:
  `org.apache.derby.jdbc.ClientConnectionPoolDataSource`
– Requires the following JDBC driver file:
  - derbyclient.jar
– Requires DataStoreHelper class:
  `com.ibm.websphere.rsadapter.DerbyNetworkServerDataStoreHelper`
– Requires the following property:

  databaseName
  The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

  If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.

- **Derby Network Server using Derby Client (XA)**

  Use this provider to access only Apache Derby databases that run on the same node as the application server.

  This provider:

  – Does not support Version 4.0 data sources.
  – Is configurable only in nodes at version 6.1 and later
  – Uses the following XA data source for this Derby Network Server using Derby Client provider:
    `org.apache.derby.jdbc.ClientXADataSource`
  – Requires the following JDBC driver file:
    - derbyclient.jar
  – Requires the DataStoreHelper class:
    `com.ibm.websphere.rsadapter.DerbyNetworkServerDataStoreHelper`
  – Requires the following property:

    databaseName
    The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

    If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.

- **Derby Network Server using Derby Client 40**

  Use this provider to access only Apache Derby databases that run on the same node as the application server.

  This provider:

  – Is configurable only in nodes at version 7.0 and later
  – Does not support Version 4.0 data sources.
  – Uses the following one phase data source for the Derby Network Server using Derby Client provider:
    `org.apache.derby.jdbc.ClientConnectionPoolDataSource40`
  – Requires the following JDBC driver file:
    - derbyclient.jar

– Requires DataStoreHelper class:
  `com.ibm.websphere.rsadapter.DerbyNetworkServerDataStoreHelper`
– Requires the following property:

  databaseName
  The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

  If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.

- **Derby Network Server using Derby Client 40 (XA)**

  Use this provider to access only Apache Derby databases that run on the same node as the application server.

  This provider:
  – Is configurable only in nodes at version 7.0 and later
  – Does not support Version 4.0 data sources.
  – Uses the following one phase data source for the Derby Network Server using Derby Client provider:
    `org.apache.derby.jdbc.ClientXADataSource40`
  – Requires the following JDBC driver file:
    - derbyclient.jar
  – Requires DataStoreHelper class:
    `com.ibm.websphere.rsadapter.DerbyNetworkServerDataStoreHelper`
  – Requires the following property:

    databaseName
    The name of the database from which the data source obtains connections. If you do not specify a fully qualified path name, the application server uses the default location of *app_server_root*/derby or the equivalent default for AIX, HP-UX, Linux, or Solaris system environments. An example for the database path name is:

    If no database exists for the path name that you want to specify, configure the custom property, `createDatabase`, to a value of `create` to create the database dynamically.

Data source minimum required settings for DB2 Universal Database for IBM i:

These properties vary according to the database vendor requirements for JDBC driver implementations. You must set the appropriate properties on every data source that you configure. These settings are for a DB2 UDB data source.

What type of configuration do you have?

- "DB2 UDB for iSeries with the application server for AIX, HP-UX, IBM i, Linux, Solaris, or Windows"

DB2 UDB for iSeries with the application server for AIX, HP-UX, IBM i, Linux, Solaris, or Windows

You can configure the following types of providers:

- DB2 UDB for iSeries (Native)
- DB2 UDB for iSeries (Native XA)
- DB2 UDB for iSeries (Toolbox)
- DB2 UDB for iSeries (Toolbox XA)
- **DB2 UDB for iSeries (Native)**

The iSeries Developer Kit for Java contains this Type 2 JDBC driver that is built on top of the iSeries DB2 Call Level Interface (CLI) native libraries.

This provider:

– Is for local DB2 connections on iSeries. It is not recommended for remote access.
– Supports the one-phase data source:

  `com.ibm.db2.jdbc.app.UDBConnectionPoolDataSource`
– Requires the following JDBC driver files:
  - db2_classes16.jar - for nodes that are running at Version 7.0 or later. The location of the jar file is `/QIBM/Proddata/java400/jdk6/lib/ext/db2_classes16.jar`.
  - db2_classes.jar - for nodes that are running at Version 6.1 or earlier. The location of the jar file is `/QIBM/ProdData/Java400/ext/db2_classes.jar`.
– Requires the following DataStoreHelper class:

  `com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper`
– Does not require an authentication alias.
– Requires the following properties:
  - **databaseName** - The name of the relational database to which the data source connections are established. This name must appear in the iSeries Relational Database Directory. The default is *LOCAL.

• **DB2 UDB for iSeries (Native XA)**

The iSeries Developer Kit for Java contains this XA-compliant Type 2 JDBC driver built on top of the iSeries DB2 Call Level Interface (CLI) native libraries.

This provider:

– Is for local DB2 connections on iSeries. It is not recommended for remote access.
– Supports the following two-phase data source:

  `com.ibm.db2.jdbc.app.UDBXADataSource`
– Requires the following JDBC driver files:
  - db2_classes16.jar - for nodes that are running at Version 7.0 or later. The location of the jar file is `/QIBM/Proddata/java400/jdk6/lib/ext/db2_classes16.jar`.
  - db2_classes.jar - for nodes that are running at Version 6.1 or earlier. The location of the jar file is `/QIBM/ProdData/Java400/ext/db2_classes.jar`.
– Requires the following DataStoreHelper class:

  `com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper`
– Does not require an authentication alias.
– Requires the following properties:
  - **databaseName** - The name of the relational database to which the data source connections are established. This name must appear in the iSeries Relational Database Directory. The default is *LOCAL.

• **DB2 UDB for iSeries (Toolbox)**

This JDBC driver, also known as iSeries Toolbox driver for Java, is provided in the DB2 for iSeries database server.

This provider:

– Is for remote DB2 connections on iSeries. Use this driver instead of the IBM Developer Kit for Java JDBC Driver to access remote DB2 UDB for iSeries systems.
– Supports the following one-phase data source:

  `com.ibm.as400.access.AS400JDBCConnectionPoolDataSource`
– Requires the following JDBC driver files:
  - jt400.jar
– Requires the following DataStoreHelper class:

com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper

– Does not require an authentication alias if the application server and DB2 UDB for iSeries are installed in the same server. If they are installed in different servers, the user ID and password are required.
– Requires the following properties:
  - **serverName** - The name of the server from which the data source obtains connections. Example: *myserver.mydomain.com*.

- **DB2 UDB for iSeries (Toolbox XA)**

  This XA compliant JDBC driver, also known as iSeries Toolbox XA compliant driver for Java, is provided in the DB2 for iSeries database server.

  This provider:

  – Is for remote DB2 connections on iSeries. Use this driver instead of the IBM Developer Kit for Java JDBC Driver to access remote DB2 UDB for iSeries systems.
  – Supports the following two-phase data source:

    com.ibm.as400.access.AS400JDBCXADataSource

  – Requires the following JDBC driver files:
    - jt400.jar
  – Requires the following DataStoreHelper class:

    com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper

  – Does not require an authentication alias if the application server and DB2 UDB for iSeries are installed in the same server. If they are installed in different servers, the user ID and password are required.
  – Requires the following properties:
    - **serverName** - The name of the server from which the data source obtains connections. Example: *myserver.mydomain.com*.

Data source minimum required settings for Informix:

These properties vary according to the database vendor requirements for JDBC driver implementations. You must set the appropriate properties on every data source that you configure. These settings are for Informix data sources.

You can configure the following types of providers:

- Informix JDBC Driver
- Informix JDBC Driver (XA)
- Informix Using IBM JCC Driver
- Informix Using IBM JCC Driver (XA)
- Informix Using IBM DB2 JDBC Universal Driver
- Informix Using IBM DB2 JDBC Universal Driver (XA)
- **Informix JDBC Driver**

  The Informix JDBC Driver is a Type 4 JDBC driver that provides JDBC access to the Informix database.

  Informix JDBC Driver supports one phase data source:

  com.informix.jdbcx.IfxConnectionPoolDataSource

  Requires the following JDBC driver files:

  ifxjdbc.jar
  ifxjdbcx.jar

  Requires the following DataStoreHelper class:

  com.ibm.websphere.rsadapter.InformixDataStoreHelper

Requires a valid authentication alias.

Requires the following properties:

– **serverName**

The name of the Informix instance on the server. Example: ol_myserver.

– **portNumber**

The port on which the instances listen. Example: 1526.

– **ifxIFXHOST**

Either the IP address or the host name of the machine that is running the Informix database to which you want to connect. Example: myserver.mydomain.com.

To support IPv6: On AIX and Solaris, IBM Informix Dynamic Server 10.00 with fix pack 1 supports the IPv6 standard. To enable IPv6 on your WebSphere Application Server connection with one of these Informix releases, input your full IPv6 host name for the ifxIFXHOST property.

– **databaseName**

The name of the database from which the data source obtains connections. Example: Sample.

– **informixLockModeWait**

Although not required, this property enables you to set the number of seconds that Informix software waits for a lock. By default, Informix code throws an exception if it cannot immediately acquire a lock. Example: 2.

• **Informix JDBC Driver (XA)**

The Informix JDBC Driver (XA) is a Type 4 JDBC driver that provides XA-compliant JDBC access to the Informix database.

Informix JDBC Driver (XA) supports two phase data source:

```
com.informix.jdbcx.IfxXADataSource
```

Requires the following JDBC driver files:

```
ifxjdbc.jar
ifxjdbcx.jar
```

To use SQLJ: This provider also requires driver file ifxsqlj.jar if you plan to use SQLJ for queries.

Requires the following DataStoreHelper class:

```
com.ibm.websphere.rsadapter.InformixDataStoreHelper
```

Requires a valid authentication alias.

Requires the following properties:

– **serverName**

The name of the Informix instance on the server. Example: ol_myserver.

– **portNumber**

The port on which the instances listen. Example: 1526.

– **ifxIFXHOST**

Either the IP address or the host name of the machine that is running the Informix database to which you want to connect. Example: myserver.mydomain.com.

To support IPv6: On AIX and Solaris, IBM Informix Dynamic Server 10.00 with fix pack 1 supports the IPv6 standard. To enable IPv6 on your WebSphere Application Server connection with one of these Informix releases, input your full IPv6 host name for the ifxIFXHOST property.

– **databaseName**

The name of the database from which the data source obtains connections. Example: Sample.

– **ifxIFX_XASPEC**

Turn on this property when multiple users access the same database. Activating the property enforces tight coupling of XA transactions within the same global transaction ID, and requires the transactions to share lock space.

These parameters help prevent transaction management errors from occurring in the case of multiple client requests.

Turn on the ifxIFX_XASPEC property by assigning it the value of `Y` or `y`; either character works because the setting is not case-specific. Turn the property off by assigning it the value of `N` or `n`. WebSphere Application Server ignores all other values. Your setting for the property overrides the Informix database system setting.

– **informixLockModeWait**

Although not required, this property enables you to set the number of seconds that Informix software waits for a lock. By default, Informix code throws an exception if it cannot immediately acquire a lock. Example: 2.

- **Informix Using IBM JCC Driver**

The Informix IBM JCC Driver is a one-phase commit provider for Informix that uses the IBM Data Server Driver for JDBC and SQLJ. The IBM Data Server Driver is JDBC 4.0 compliant and is the next generation of the Universal JCC driver.

This provider is configurable in version 7.0 and later nodes.

The following one-phase data source is supported:

```
com.ibm.db2.jcc.DB2ConnectionPoolDataSource
```

The following JDBC driver files are required:

```
db2jcc4.jar
db2jcc_license_cu.jar
db2jcc_license_cisuz.jar
```

as well as the following DataStoreHelper class:

```
com.ibm.websphere.rsadapter.InformixJccDataStoreHelper
```

This provider requires a valid authentication alias.

The following properties are required:

– **serverName** - The TCP/IP address or host name for the Informix server.

– **portNumber** - The TCP/IP port number where the Informix server resides.

– **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.

- **Informix Using IBM JCC Driver (XA)**

The Informix IBM JCC Driver (XA) is a two-phase commit provider for Informix that uses the IBM Data Server Driver for JDBC and SQLJ. The IBM Data Server Driver is JDBC 4.0 compliant and is the next generation of the Universal JCC driver.

This provider is configurable in version 7.0 and later nodes.

The following two-phase data source is supported:

```
com.ibm.db2.jcc.DB2XADataSource
```

The following JDBC driver files are required:

```
db2jcc4.jar
db2jcc_license_cu.jar
db2jcc_license_cisuz.jar
```

as well as the following DataStoreHelper class:

```
com.ibm.websphere.rsadapter.InformixJccDataStoreHelper
```

Note: If you plan to use SQLJ for queries, this provider also requires driver file `ifxsqlj.jar`.

This provider requires a valid authentication alias.

The following properties are required:

– **serverName** - The TCP/IP address or host name for the Informix server.

– **portNumber** - The TCP/IP port number where the Informix server resides.

– **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.

Note: You cannot use Informix XA data sources with ANSI databases if SQL statements are issued in local transactions instead of global transactions. This scenario might occur within the application code or within a component of Application Server such as scheduler. The following message might be logged if you are experiencing this problem:

```
java.sql.SQLException: Already in transaction.
       at com.informix.util.IfxErrMsg.getSQLException(IfxErrMsg.java:398)
       at com.informix.jdbc.IfxSqli.a(IfxSqli.java:3247)
       at com.informix.jdbc.IfxSqli.E(IfxSqli.java:3556)
       at com.informix.jdbc.IfxSqli.dispatchMsg(IfxSqli.java:2382)
       at com.informix.jdbcx.IfxXASqli.receiveMessage(IfxXASqli.java:120)
       at com.informix.jdbc.IfxSqli.X(IfxSqli.java:7926)
       at com.informix.jdbc.IfxSqli.a(IfxSqli.java:854)
       at com.informix.jdbc.IfxSqli.executeCommand(IfxSqli.java:749)
       at com.informix.jdbc.IfxResultSet.b(IfxResultSet.java:293)
       at com.informix.jdbc.IfxStatement.c(IfxStatement.java:1269)
       at com.informix.jdbc.IfxStatement.b(IfxStatement.java:423)
       at com.informix.jdbc.IfxStatement.executeUpdate(IfxStatement.java:277)
       at com.informix.jdbc.IfxSqliConnect.setTransactionIsolation(IfxSqliConnect.java:2565)
```

To avoid this issue:

– Switch to a non-ANSI database.

– If the error is triggered by an application, update the application such that it always runs in a global transaction.

- **Informix Using IBM DB2 JDBC Universal Driver**

  The Informix JDBC Driver is a Type 4 JDBC driver that is JDBC 3.0 compliant and provides access to the Informix database.

  This provider supports the following one-phase data source:

  ```
  com.ibm.db2.jcc.DB2ConnectionPoolDataSource
  ```

  The following JDBC driver files are required:

  ```
  db2jcc.jar
  db2jcc_license_cu.jar
  db2jcc_license_cisuz.jar
  ```

  as well as the following DataStoreHelper class:

  ```
  com.ibm.websphere.rsadapter.InformixJccDataStoreHelper
  ```

  This provider requires a valid authentication alias.

  The following properties are required:

  – **serverName** - The TCP/IP address or host name for the Informix server.

  – **portNumber** - The TCP/IP port number where the Informix server resides.

  – **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.

- **Informix Using IBM DB2 JDBC Universal Driver (XA)**

  The Informix Using JDBC Driver (XA) is a Type 4 JDBC driver that is JDBC 3.0 compliant and provides XA-compliant JDBC access to the Informix database.

  This provider supports the following two-phase data source:

  ```
  com.ibm.db2.jcc.DB2XADataSource
  ```

  The following JDBC driver files are required:

  ```
  db2jcc.jar
  db2jcc_license_cu.jar
  db2jcc_license_cisuz.jar
  ```

  as well as the following DataStoreHelper class:

  ```
  com.ibm.websphere.rsadapter.InformixJccDataStoreHelper
  ```

This provider requires a valid authentication alias.

The following properties are required:

– **serverName** - The TCP/IP address or host name for the Informix server.
– **portNumber** - The TCP/IP port number where the Informix server resides.
– **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.

Note: You cannot use Informix XA data sources with ANSI databases if SQL statements are issued in local transactions instead of global transactions. This scenario might occur within the application code or within a component of Application Server such as scheduler. The following message might be logged if you are experiencing this problem:

```
java.sql.SQLException: Already in transaction.
       at com.informix.util.IfxErrMsg.getSQLException(IfxErrMsg.java:398)
       at com.informix.jdbc.IfxSqli.a(IfxSqli.java:3247)
       at com.informix.jdbc.IfxSqli.E(IfxSqli.java:3556)
       at com.informix.jdbc.IfxSqli.dispatchMsg(IfxSqli.java:2382)
       at com.informix.jdbcx.IfxXASqli.receiveMessage(IfxXASqli.java:120)
       at com.informix.jdbc.IfxSqli.X(IfxSqli.java:7926)
       at com.informix.jdbc.IfxSqli.a(IfxSqli.java:854)
       at com.informix.jdbc.IfxSqli.executeCommand(IfxSqli.java:749)
       at com.informix.jdbc.IfxResultSet.b(IfxResultSet.java:293)
       at com.informix.jdbc.IfxStatement.c(IfxStatement.java:1269)
       at com.informix.jdbc.IfxStatement.b(IfxStatement.java:423)
       at com.informix.jdbc.IfxStatement.executeUpdate(IfxStatement.java:277)
       at com.informix.jdbc.IfxSqliConnect.setTransactionIsolation(IfxSqliConnect.java:2565)
```

Data source minimum required settings for Microsoft SQL Server:

These properties vary according to the database vendor requirements for Java Database Connectivity (JDBC) driver implementations. You must set the appropriate properties on every data source that you configure. These settings are for Microsoft SQL Server data sources.

The application server also supports two options for setting isolation level in Microsoft SQL Server: SNAPSHOT and READ_COMMITTED_SNAPSHOT.

Table 28. Isolation levels in Microsoft SQL Server. The following table describes these isolation levels and configuration considerations.

JDBC provider	Microsoft SQL Server feature	Configuration consideration
Microsoft SQL Server JDBC Driver	SNAPSHOT isolation level	Set the isolation level constant by invoking the setTransactionIsolation method with one of the following attributes: • conn.setTransactionIsolation (com.microsoft.sqlserver.jdbc. SQLServerConnection. TRANSACTION_SNAPSHOT) • conn.setTransactionIsolation(*value_of_constant*)
	READ_COMMITTED_SNAPSHOT isolation level	This isolation level is an implementation of the Read committed isolation level. The policy enforces optimistic locking for read operations with Microsoft SQL Server. 1. Configure the isolation level on the database. 2. Invoke the setTransactionIsolation method with the conn.setTransactionIsolation (java.sql.Connection. TRANSACTION_READ_COMMITTED) attribute.

Table 28. Isolation levels in Microsoft SQL Server (continued). The following table describes these isolation levels and configuration considerations.

JDBC provider	Microsoft SQL Server feature	Configuration consideration
DataDirect ConnectJDBC type 4 driver for Microsoft SQL Server	SNAPSHOT isolation level	This isolation level implements optimistic locking for transactions in which Microsoft SQL Server serializes the data. Configure the ALLOW_SNAPSHOT_ISOLATION setting on the database, and then set the isolation level in one of two ways: • By isolation level constant. Invoke the setTransactionIsolation method with one of the following attributes: – conn.setTransactionIsolation (com.ddtek.jdbc.extensions. ExtConstants.TRANSACTION_SNAPSHOT) – conn.setTransactionIsolation(16) • By the custom data source property: – Set the data source custom property snapshotSerializable to `true`. – Invoke the setTransactionIsolation method with the conn.setTransactionIsolation (java.sql.Connection. TRANSACTION_SERIALIZABLE) attribute:
	READ_COMMITTED_SNAPSHOT isolation level	This isolation level is an implementation of the Read committed isolation level. The policy enforces optimistic locking for read operations with Microsoft SQL Server. 1. Configure the isolation level on the database. 2. Invoke the setTransactionIsolation method with the conn.setTransactionIsolation (java.sql.Connection. TRANSACTION_READ_COMMITTED) attribute.

Consult the Backward Compatibility for Microsoft SQL Server components web page for a complete list of deprecated items, as well as backward compatibility provisions, for Microsoft SQL Server.

You can configure the following types of providers:

- Microsoft SQL Server JDBC Driver
- Microsoft SQL Server JDBC Driver (XA)
- DataDirect ConnectJDBC type 4 driver for Microsoft SQL Server
- DataDirect ConnectJDBC type 4 driver for Microsoft SQL Server (XA)
- **Microsoft SQL Server JDBC Driver**

  The Microsoft SQL Server JDBC driver supports this data source:

  `com.microsoft.sqlserver.jdbc.SQLServerConnectionPoolDataSource`

  The JDBC provider requires the following Java archive (JAR) files:

  `sqljdbc4.jar`

  The JDBC provider requires the following DataStoreHelper class:

  `com.ibm.websphere.rsadapter.MicrosoftSQLServerDataStoreHelper`

  The JDBC provider requires a valid authentication alias.

  The JDBC driver requires the following properties:

  serverName
  Specifies the name of the server in which Microsoft SQL Server resides. Example: myserver.mydomain.com

  portNumber
  Specifies the TCP/IP port that Microsoft SQL Server uses for communication. Port 1433 is the default.

databaseName
Specifies the name of the database from which the data source obtains connections. Example: Sample.

- **Microsoft SQL Server JDBC Driver (XA)**

  This JDBC provider supports this data source:

  com.microsoft.sqlserver.jdbc.SQLServerXADataSource

  The JDBC provider requires the following Java archive (JAR) files:

  sqljdbc4.jar

  The JDBC provider requires the following DataStoreHelper class:

  com.ibm.websphere.rsadapter.MicrosoftSQLServerDataStoreHelper

  The JDBC provider requires a valid authentication alias.

  The JDBC driver requires the following properties:

  serverName
  Specifies the name of the server in which Microsoft SQL Server resides. Example: myserver.mydomain.com

  portNumber
  Specifies the TCP/IP port that Microsoft SQL Server uses for communication. Port 1433 is the default.

  databaseName
  Specifies the name of the database from which the data source obtains connections. Example: Sample.

- **DataDirect ConnectJDBC type 4 driver for Microsoft SQL Server**

  DataDirect ConnectJDBC type 4 driver for Microsoft SQL Server is a Type 4 JDBC driver that provides JDBC access to the Microsoft SQL Server databases. This provider is for use only with the Connect JDBC driver purchased from DataDirect Technologies.

  This JDBC provider supports the following data source:

  com.ddtek.jdbcx.sqlserver.SQLServerDataSource

  Requires JDBC driver files:

  sqlserver.jar

  Requires **DataStoreHelper** class:

  com.ibm.websphere.rsadapter.ConnectJDBCDataStoreHelper

  Requires a valid authentication alias.

  Requires properties:

  – **serverName** The name of the server in which Microsoft SQL Server resides. Example: *myserver.mydomain.com*
  – **portNumber** The TCP/IP port that Microsoft SQL Server uses for communication. Port 1433 is the default.
  – **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

- **DataDirect ConnectJDBC type 4 driver for Microsoft SQL Server (XA)**

  DataDirect ConnectJDBC type 4 driver for Microsoft SQL Server (XA) is a Type 4 JDBC driver which provides XA-compliant JDBC access to the Microsoft SQL Server databases. This provider is for use only with the Connect JDBC driver purchased from DataDirect Technologies.

  This JDBC provider supports this data source:

  com.ddtek.jdbcx.sqlserver.SQLServerDataSource.

  Requires JDBC driver files:

  sqlserver.jar

  Requires **DataStoreHelper** class:

  com.ibm.websphere.rsadapter.ConnectJDBCDataStoreHelper

  Requires a valid authentication alias.

Requires properties:
– **serverName** The name of the server in which Microsoft SQL Server resides. Example: *myserver.mydomain.com*
– **portNumber** The TCP/IP port that Microsoft SQL Server uses for communication. Port 1433 is the default.
– **databaseName** The name of the database from which the data source obtains connections. Example: *Sample*.

Data source minimum required settings for Oracle:

These properties vary according to the database vendor requirements for JDBC driver implementations. You must set the appropriate properties on every data source that you configure. These settings are for Oracle data sources.

You can configure the following types of providers:
- Oracle JDBC Driver
- Oracle JDBC Driver (XA)
- **Oracle JDBC Driver**

  The Oracle JDBC Driver provides JDBC access to the Oracle database. This JDBC driver supports both Type 2 JDBC access and Type 4 JDBC access.

  This provider:
  – Supports one-phase data source:

    `oracle.jdbc.pool.OracleConnectionPoolDataSource`
  – Requires the following JDBC driver files:
    - ojdbc6.jar.

      Note: Be aware of the following:
      - Oracle does not support the use of `ojdbc5.jar` in an environment with the Java SE Development Kit Version 6 or later.
      - In mixed node environments, the data source wizard in the administrative console allows you to choose a class path for `ojdbc6.jar` or `ojdbc5.jar`.
      - For Oracle trace, use `ojdbc`*`version`*`_g.jar`.
  – Requires the following DataStoreHelper class:

    `com.ibm.websphere.rsadapter.Oracle11gDataStoreHelper`

    Note: You must use the Oracle11gDataStoreHelper with the `ojdbc6.jar` driver file, regardless of whether you use an Oracle 11g or Oracle 10g database server.
  – Requires a valid authentication alias.
  – Requires properties:

    URL The URL that indicates the database from which the data source obtains connections. For example:

    `jdbc:oracle:thin:@//`*`myServer`*`:1521/`*`myDatabase`*

    where *myServer* is the server name, *1521* is the port that the server uses for communication, and *myDatabase* is the database name.
- **Oracle JDBC Driver (XA)**

  The Oracle JDBC Driver (XA) provides XA-compliant JDBC access to the Oracle database. This JDBC driver supports both Type 2 JDBC access and Type 4 JDBC access.

  This provider:
  – Supports two-phase data source:

    `oracle.jdbc.xa.client.OracleXADataSource`
  – Requires the following JDBC driver files:
    - `ojdbc6.jar`

Note: Be aware of the following notes regarding the use of these driver files:
- Oracle does not support the use of `ojdbc5.jar` in an environment with the Java SE Development Kit Version 6 or later.
- In mixed node environments, the data source wizard in the administrative console allows you to choose a class path for `ojdbc6.jar` or `ojdbc5.jar`.
- For Oracle trace, use `ojdbc`*`version`*`_g.jar`.

– Requires the following DataStoreHelper class:

  `com.ibm.websphere.rsadapter.Oracle11gDataStoreHelper`

  Note: You must use the Oracle11gDataStoreHelper with the `ojdbc6.jar` driver file, regardless of whether you use an Oracle 11g or Oracle 10g database server.
– Requires a valid authentication alias.
– Requires properties:

  URL Indicates the database from which the data source obtains connections. For example:

  `jdbc:oracle:thin:@//`*`myServer`*`:1521/`*`myDatabase`*

  where *myServer* is the server name, *1521* is the port that the server uses for communication, and *myDatabase* is the database name.

Data source minimum required settings for Sybase:

These properties vary according to the database vendor requirements for JDBC driver implementations. You must set the appropriate properties on every data source that you configure. These settings are for Sybase data sources.

What types of providers can you configure?

You can configure the following types of providers:
- Sybase JDBC 4 Driver
- Sybase JDBC 4 Driver (XA)
- Sybase JDBC 3 Driver
- Sybase JDBC 3 Driver (XA)
- Sybase JDBC 2 Driver
- Sybase JDBC 2 Driver (XA)

Sybase JDBC 4 Driver

The Sybase JDBC 4 Driver is a Type 4 JDBC driver that provides JDBC access to the Sybase database.

This provider:
- Uses jConnect Version 7.0
- Supports one phase data source:

  `com.sybase.jdbc4.jdbc.SybConnectionPoolDataSource`
- Requires the following JDBC driver files:

  `jconn4.jar`
- Requires the following DataStoreHelper class:

  `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`
- Requires a valid authentication alias.
- Requires the following properties:
  – **serverName** - The name of the database server. Example: *myserver.mydomain.com*.
  – **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.
  – **portNumber** - The TCP/IP port number through which all communications to the server take place. Example: *5000*.

– **connectionProperties** - A custom property required for applications containing EJB 2.0 enterprise beans. Value: `SELECT_OPENS_CURSOR=true` (Type: java.lang.String)

Sybase JDBC 4 Driver (XA)

The Sybase JDBC 4 Driver (XA) is a Type 4 JDBC driver that provides XA-compliant JDBC access to the Sybase database.

This provider:
- Uses jConnect Version 7.0
- Supports two phase data source:
  `com.sybase.jdbc4.jdbc.SybXADataSource`
- Requires JDBC driver files:
  `jconn4.jar`
- Requires the following DataStoreHelper class:
  `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`
- Requires a valid authentication alias.
- Requires the following properties:
  – **serverName** - The name of the database server. Example: *myserver.mydomain.com*
  – **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.
  – **portNumber** - The TCP/IP port number through which all communications to the server take place. Example: *5000*.
  – **connectionProperties** - A custom property required for applications containing EJB 2.0 enterprise beans. Value: `SELECT_OPENS_CURSOR=true` (Type: java.lang.String)

Sybase JDBC 3 Driver

The Sybase JDBC 3 Driver is a Type 4 JDBC driver that provides JDBC access to the Sybase database.

This provider:
- Uses jConnect Version 6.05
- Supports one phase data source:
  `com.sybase.jdbc3.jdbc.SybConnectionPoolDataSource`
- Requires the following JDBC driver files:
  `jconn3.jar`
- Requires the following DataStoreHelper class:
  `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`
- Requires a valid authentication alias.
- Requires the following properties:
  – **serverName** - The name of the database server. Example: *myserver.mydomain.com*.
  – **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.
  – **portNumber** - The TCP/IP port number through which all communications to the server take place. Example: *5000*.
  – **connectionProperties** - A custom property required for applications containing EJB 2.0 enterprise beans. Value: `SELECT_OPENS_CURSOR=true` (Type: java.lang.String)

Sybase JDBC 3 Driver (XA)

The Sybase JDBC 3 Driver (XA) is a Type 4 JDBC driver that provides XA-compliant JDBC access to the Sybase database.

This provider:
- Uses jConnect Version 6.05

- Supports two phase data source:
  `com.sybase.jdbc3.jdbc.SybXADataSource`
- Requires JDBC driver files:
  `jconn3.jar`
- Requires the following DataStoreHelper class:
  `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`
- Requires a valid authentication alias.
- Requires the following properties:
  - **serverName** - The name of the database server. Example: *myserver.mydomain.com*
  - **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.
  - **portNumber** - The TCP/IP port number through which all communications to the server take place. Example: *5000*.
  - **connectionProperties** - A custom property required for applications containing EJB 2.0 enterprise beans. Value: `SELECT_OPENS_CURSOR=true` (Type: java.lang.String)

Sybase JDBC 2 Driver

The Sybase JDBC 2 Driver is a Type 4 JDBC driver that provides JDBC access to the Sybase database.

This provider:
- Uses jConnect Version 5.5
- Supports one phase data source:
  `com.sybase.jdbc2.jdbc.SybConnectionPoolDataSource`
- Requires the following JDBC driver files:
  `jconn2.jar`
- Requires the following DataStoreHelper class:
  `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`
- Requires a valid authentication alias.
- Requires the following properties:
  - **serverName** - The name of the database server. Example: *myserver.mydomain.com*.
  - **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.
  - **portNumber** - The TCP/IP port number through which all communications to the server take place. Example: *5000*.
  - **connectionProperties** - A custom property required for applications containing EJB 2.0 enterprise beans. Value: `SELECT_OPENS_CURSOR=true` (Type: java.lang.String)

Sybase JDBC 2 Driver (XA)

The Sybase JDBC 2 Driver (XA) is a Type 4 JDBC driver that provides XA-compliant JDBC access to the Sybase database.

This provider:
- Uses jConnect Version 5.5
- Supports two phase data source:
  `com.sybase.jdbc2.jdbc.SybXADataSource`
- Requires JDBC driver files:
  `jconn2.jar`
- Requires the following DataStoreHelper class:
  `com.ibm.websphere.rsadapter.SybaseDataStoreHelper`
- Requires a valid authentication alias.
- Requires the following properties:
  - **serverName** - The name of the database server. Example: *myserver.mydomain.com*

– **databaseName** - The name of the database from which the data source obtains connections. Example: *Sample*.
– **portNumber** - The TCP/IP port number through which all communications to the server take place. Example: *5000*.
– **connectionProperties** - A custom property required for applications containing EJB 2.0 enterprise beans. Value: `SELECT_OPENS_CURSOR=true` (Type: java.lang.String)

Configuring a JDBC provider using the administrative console

To create connections between an application and a relational database, the application server uses the driver implementation classes that are encapsulated by the Java Database Connectivity (JDBC) provider.

Before you begin

Each JDBC provider is essentially an object that represents vendor-specific JDBC driver classes to the application server, for establishing access to that particular vendor database. JDBC providers are prerequisites for data sources, which supply applications with the physical connections to a database. Consult the JDBC provider table to identify the appropriate JDBC provider for your database and application requirements.

About this task

Configure at least one JDBC provider for each database server that you plan to use at a particular scope within your application server environment.

Procedure

1. Open the administrative console.
2. Click **Resources** > **JDBC** > **JDBC Providers**.
3. Select the scope at which applications can use the JDBC provider. The scope that you select becomes the scope of any data source that you associate with this provider. You can choose a cell, node, cluster, or server. For more information about scope and how it can affect resources, see the information center topic on administrative scope settings.
4. Click **New**. This action causes the **Create a new JDBC Provider** wizard to launch.
5. Use the first drop-down list to select the database type of the JDBC provider that you must create.

   The User-Defined option: Select **User-Defined** for your database type if you encounter either of the following scenarios:
   - You do not see your database type.
   - You cannot select the JDBC provider type that you need in the next step.

   The user-defined selection triggers the wizard panel to display your provider type as a user-defined JDBC provider, and your implementation type as user-defined. Consult your database documentation for the JDBC driver class files, data source properties, and so on, that are required for your user-defined provider. You must supply this information about the next two panels:
   - database class path
   - database-specific properties
6. Select your JDBC provider type if it is displayed in the second drop-down list. Select **Show Deprecated** to trigger the display of both current and deprecated providers. If you cannot find your provider in this expanded list, then select **User-Defined** from the previous list of database types.
7. From the third drop-down list, select the implementation type that is necessary for your application. If your application does not require that connections support two-phase commit transactions, choose **Connection Pool Data Source**. Choose **XA Data Source**, however, if your application requires

connections that support two-phase commit transactions. Applications that use this data source configuration have the benefit of container-managed transaction recovery.

After you select an implementation type, the wizard fills the name and the description fields for your JDBC provider. You can type different values for these fields; they exist for administrative purposes only.

8. Click **Next** to see the **Enter database class path information** wizard panel.
9. In the class path field, type the full path location of the database JDBC driver class files. Your class path information becomes the value of the WebSphere environment variable that is displayed on this panel, in the form of ${*DATABASE*_JDBC_DRIVER_PATH}. The application server uses the variable to define your JDBC provider; this practice eliminates the must specify static JDBC class paths for individual applications. Remember that if you do not provide the full, correct JDBC driver class path for the variable, your data source ultimately fails. If the field already displays a fully qualified class path, you can accept that variable definition by completing the rest of this wizard panel and clicking **Next**.

   Note: The application server supports multiple versions of the selected JDBC driver for the DataDirect ConnectJDBC type 4 driver for MS SQL Server. Each version of the JDBC driver has a unique class path. Select the appropriate version of the JDBC driver so the class path is populated correctly.
10. Use the Native library path field to specify additional class files that your JDBC driver might require to function properly on your application server platform. Type the full directory path name of these class files.

    gotcha: If you are using an Oracle OCI driver as your JDBC provider, you must specify the path to where the native libraries are stored. If you do not specify a native library path, the first time you try to connect using this provider, class loader errors occur.
11. Click **Next** to see a summary of your JDBC provider settings.
12. Click **Finish** if you are satisfied with the JDBC provider configuration. You now see the JDBC provider collection panel, which displays your new JDBC provider in a table along with other providers that are configured for the same scope.

What to do next

The next step is to create a data source to associate with your JDBC provider. For detailed information, see the information center topic on configuring a data source using the administrative console.

Remember: If you modify configuration of a JDBC provider, like the class path, native library path, or custom properties, click **OK** and then restart every application server within the scope of that JDBC provider. Otherwise, the new configuration does not work and you receive data source failure messages.

JDBC provider collection:

Use this page to view JDBC providers. The JDBC provider object encapsulates the specific JDBC driver implementation class for the data sources that you define and associate with the provider.

To view this administrative console page, click **Resources** > **JDBC** > **JDBC providers** .

Name:

Specifies a text identifier for this provider.

For example, this field can be *DB2 JDBC Provider (XA)*.

Data type String

Scope:

Specifies the scope of the JDBC provider; if you use any scope other than the default of *Node*, the provider might not be available in other scope contexts. Data sources that are created with this JDBC provider inherit this scope.

Description:

Specifies a text string describing this provider.

Data type String

JDBC provider settings:

Use this page to modify the settings for a JDBC provider.

To view this administrative console page, click **Resources** > **JDBC** > **JDBC providers** > ***JDBC_provider***.

Important: If you use this page to modify the class path or native library path of an existing JDBC provider: After you apply and save the new settings, you must restart every application server within the scope of that JDBC provider for the new configuration to work. Otherwise, you receive a data source failure message.

Scope:

Specifies the scope of the JDBC provider; data sources that are created with this JDBC provider inherit this scope.

Name:

Specifies the name of the resource provider.

Data type String

Description:

Specifies a text description for the resource provider.

Data type String

Class path:

Specifies a list of paths or JAR file names which together form the location for the resource provider classes.

For example:

- *QIBM/ProdData/Java400/ext/db2_classes.jar* for iSeries platforms.

Class path entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Class paths contain variable (symbolic) names which you can substitute using a variable map. Check the driver installation notes for the specific required JAR file names.

Data type String

Native Library Path:

Specifies a list of paths that forms the location for the resource provider native libraries.

Native path entries are separated by using the ENTER key and must not contain path separator characters (such as ';' or ':'). Native paths can contain variable (symbolic) names which you can substitute using a variable map.

Data type String

Isolate this resource provider:

Specifies that this resource provider will be loaded in its own class loader. This allows different versions or implementations of the same resource provider to be loaded in the same Java Virtual Machine. Give each version of the resource provider a unique class path that is appropriate for that version or implementation.

Note: Be aware of the following:

- You cannot isolate a resource provider if you specify a native library path. The Application Server will define a value for the native library path for some JDBC providers; this behavior is intended to help you configure your provider when a native library path is necessary. If you do not require the native library path, delete the value, and you will be able to select the option to isolate the resource provider.
- If you are running a mixed cell environment, the application server will remove any isolated JDBC providers from nodes that are running at versions earlier than 7.0 if the provider is scoped for a version 7.0 cell, and you have not migrated the provider from an older release. If you want to use isolated resources at the cell level, do not use the resources in nodes that are running at versions earlier than 7.0. Define a resource at the node level, or avoid using the resource in nodes that are earlier than version 7.0, because this will result in a "Naming not found" exception when the application server attempts to perform a lookup on an isolated resource at the cell level.

Implementation class name:

Specifies the Java class name of the JDBC driver implementation.

This class is available in the driver file mentioned in the class path description above.

For example, *com.ibm.db2.jdbc.app.UDBXADataSource* for iSeries platforms.

Note: If you modify the implementation class name of the JDBC provider after you have created the provider, you might disconnect the provider from the template used to create it. As a result, data sources created from this JDBC provider do not have an associated template; you must manually configure a working data source through setting custom properties.

Data type String

JDBC provider summary:

JDBC providers are prerequisites for data sources, which supply applications with the physical connections to a database.

Use these tables for quick reference on database-specific JDBC providers.

Table 29. Apache Derby JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Apache Derby		
JDBC provider	**Transaction support**	**Version and other considerations**
Derby JDBC Provider	One-phase	• Does not support Version 4 data sources • Configurable only in nodes at version 6.0.2 and later • Not for use in clustered environment: accessible from a single JVM only
Derby JDBC Provider (XA)	One and two phase	• Does not support Version 4 data sources • Configurable only in nodes at version 6.0.2 and later • Not for use in clustered environment: accessible from a single JVM only
Derby JDBC Provider 40	One-phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources
Derby JDBC Provider 40 (XA)	One and two phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources
Derby Network Server Using Derby Client	One-phase	• Does not support Version 4.0 data sources. • Configurable only in nodes at version 6.1 and later • *Can be used in clustered environment:* a database instance can be accessed by multiple JVMs • Only for use with Apache Derby databases that run on the same node as the application server
Derby Network Server Using Derby Client (XA)	One and two phase	• Does not support Version 4 data sources • Configurable only in nodes at version 6.1 and later • *Can be used in clustered environment:* a database instance can be accessed by multiple JVMs • Only for use with Apache Derby databases that run on the same node as the application server
Derby Network Server Using Derby Client 40	One-phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources

Table 29. Apache Derby JDBC providers (continued). Use the table for quick reference on database-specific JDBC providers.

Apache Derby		
JDBC provider	**Transaction support**	**Version and other considerations**
Derby Network Server Using Derby Client 40 (XA)	One and two phase	• Configurable only in nodes at version 7.0 and later • Does not support Version 4 data sources

Table 30. DB2 on AIX, HP-UX, Linux, Solaris, and Windows systems JDBC providers. Use the table for quick reference on database-specific JDBC providers.

DB2 on AIX, HP-UX, Linux, Solaris, and Windows systems		
JDBC provider	**Transaction support**	**Version and other considerations**
DB2 Using IBM JCC Driver	One-phase	• Configurable in nodes that are at version 7.0 and later.
DB2 Using IBM JCC Driver (XA)	One and two phase	• Configurable in nodes that are at version 7.0 and later.
DB2 Universal JDBC Provider	One-phase	N/A
DB2 Universal JDBC Provider (XA)	One and two phase	N/A

Table 31. DB2 UDB for iSeries JDBC providers. Use the table for quick reference on database-specific JDBC providers.

DB2 UDB for iSeries		
JDBC provider	**Transaction support**	**Version and other considerations**
DB2 UDB for iSeries (Native)	One-phase	Recommended when you run the application server on iSeries.
DB2 UDB for iSeries (Native XA)	One and two phase	Recommended when you run the application server on iSeries.
DB2 UDB for iSeries (Toolbox)	One-phase	N/A
DB2 UDB for iSeries (Toolbox XA)	One and two phase	N/A

Table 32. DB2 on z/OS JDBC providers. Use the table for quick reference on database-specific JDBC providers.

DB2 on z/OS		
JDBC provider	**Transaction support**	**Version and other considerations**
DB2 Using IBM JCC Driver	One-phase	Configurable in version 7.0 and later nodes.
DB2 Using IBM JCC Driver (XA)	One and two phase	Configurable version 7.0 and later nodes.
DB2 Universal JDBC Provider	One-phase when connecting to the application server that is on AIX, HP-UX, Linux, Solaris, Windows, and iSeries systems	
DB2 Universal JDBC Provider (XA)	One and two phase	

Table 33. Informix JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Informix		
JDBC provider	**Transaction support**	**Version and other considerations**
Informix Using IBM JCC Driver	One phase	This provider is configurable in nodes that are at version 7.0 and later.
Informix Using IBM JCC Driver (XA)	One and two phase	This provider is configurable in nodes that are at version 7.0 and later.
Informix JDBC Driver	One-phase	N/A
Informix JDBC Driver (XA)	One and two phase	N/A
Informix using IBM DB2 JDBC Universal Driver	One phase	This provider is configurable in nodes that are at version 7.0 and later.
Informix using IBM DB2 JDBC Universal Driver (XA)	One and two phase	This provider is configurable in nodes that are at version 7.0 and later.

Table 34. Microsoft SQL Server JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Microsoft SQL Server		
JDBC provider	**Transaction support**	**Version and other considerations**
Microsoft SQL Server JDBC Driver	One-phase	N/A
Microsoft SQL Server JDBC Driver (XA)	One and two phase	N/A
DataDirect ConnectJDBC Provider type 4 driver for MS SQL Server	One-phase	N/A
DataDirect ConnectJDBC Provider, type 4 driver, for MS SQL Server (XA)	One and two phase	N/A

Table 35. Oracle JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Oracle		
JDBC provider	**Transaction support**	**Version and other considerations**
Oracle JDBC Driver	One-phase	Must use the ojdbc6.jar driver to connect to any version of Oracle database.
Oracle JDBC Driver(XA)	One and two phase	Must use the ojdbc6.jar driver to connect to any version of Oracle database.

Table 36. Sybase JDBC providers. Use the table for quick reference on database-specific JDBC providers.

Sybase		
JDBC provider	**Transaction support**	**Version and other considerations**
Sybase JDBC 4 Driver	One-phase	jConnect v7.0
Sybase JDBC 4 Driver (XA)	One and two phase	jConnect v7.0
Sybase JDBC 3 Driver	One-phase	jConnect v6.05
Sybase JDBC 3 Driver (XA)	One and two phase	jConnect v6.05
Sybase JDBC 2 Driver	One-phase	jConnect v5.5
Sybase JDBC 2 Driver (XA)	One and two phase	jConnect v5.5

Configuring a data source using the administrative console

Application components use a data source to access connection instances to a relational database.

Before you begin

The application server supports two different versions of data source. Determine the data source for your environment according to the enterprise bean and servlet specification levels that are the basis of your applications:

- *Data sources (WebSphere Application Server Version 4)* are for use with the Enterprise JavaBeans (EJB) 1.0 specification and the Java Servlet 2.2 specification.
- Data sources of the latest standard version are for use with applications that implement the more advanced releases of these specifications.

About this task

When you create a data source, you associate it with a Java Database Connectivity (JDBC) provider that is configured for access to a specific vendor database. The application server requires both objects for your applications to make calls to that particular database and receive data from it. The data source provides connection management capabilities that physically make possible these exchanges between your applications and the database.

Procedure

1. Open the administrative console.
2. Access the necessary console panel. Use one of the following paths:
   - Click **Resources** > **JDBC** > **Data sources**.
   - Click **Resources** > **JDBC** > **Data sources (**WebSphere Application Server Version 4)
   - Click **Resources** > **JDBC** > **JDBC providers** > ***jdbc_provider*** > **Data sources**
   - Click **Resources** > **JDBC** > **JDBC providers** > ***jdbc_provider*** > **Data sources (**WebSphere Application Server Version 4) .
3. Select the scope at which applications can use the data source. You can choose a cell, node, cluster, or server. For more information, see the topic on scope settings.

   Version 4 only: From this point onward, the steps for creating WebSphere Application Server Version 4 data sources differ from the steps for creating data sources of the latest standard version. To configure a Version 4 data source:
   - Click **New** to proceed to the console panel for defining required properties.
   - On this properties panel specify values for the fields that are grouped under the heading **Configuration**. The application server requires these properties to implement your JDBC driver classes.
   - Save your configuration by clicking **OK**. You are now finished with the primary data source configuration tasks.
   - Define other properties that your database vendor might require, or offer as options, for using the JDBC driver. The application server calls them custom properties, and requires that you set them on the data source. Begin by clicking the **Custom Properties** link that is now displayed on the administrative console panel. Consult your database documentation to learn about these required and optional properties.
4. Click **New**. This action causes the **Create a data source** wizard to launch and display the **Enter basic data source information** panel. The first field is the scope field, which is read-only. This field displays your previous scope selection.
5. Type a data source name in the **Data source name** field. This name identifies the data source for administrative purposes only.

6. Type a Java Naming and Directory Interface (JNDI) name in the **JNDI name** field. The application server uses the JNDI name to bind resource references for an application to this data source. Follow these requirements when you specify JNDI names:
   - Do not assign duplicate JNDI names across different resource types, such as data sources versus J2C connection factories or JMS connection factories.
   - Do not assign duplicate JNDI names for multiple resources of the same type in the same scope.

   For more information on JNDI, consult the topic on naming.
7. Click **Next** to see the **Select JDBC provider** panel. The **Select JDBC provider** panel is skipped if you do not have any JDBC providers that are configured at the current scope.
8. Select an existing JDBC provider, or create a new provider.
   - Select an existing JDBC provider.
     a. Click **Select an existing JDBC provider**.
     b. Select a JDBC driver from the drop-down list.
     c. Click **Next**. You now see the panel entitled **Enter database specific properties for the data source**.
   - Create a new JDBC provider.
     a. Click **Create new JDBC provider**.
     b. Click **Next** to see the **Create JDBC provider** panel.
     c. Use the first drop-down list to select the database type of the JDBC provider that you need to create.

        The User-Defined option: Select **User-Defined** for your database type if you encounter either of the following scenarios:
        – You do not see your database type.
        – You cannot select the JDBC provider type that you need in the next step.

        The user-defined selection triggers the wizard panel to display your provider type as a User-defined JDBC provider, and your implementation type as User-defined. Consult your database documentation for the JDBC driver class files, data source properties, and so on that are required for your user-defined provider. You must supply this information on the next two wizard panels:
        – database class path information
        – database-specific properties

     d. If the JDBC provider type is displayed in the second drop-down list, select your JDBC provider type. Select **Show Deprecated** to trigger the display of both current and deprecated providers. If you cannot find your provider in this expanded list, then select **User-Defined** from the previous list of database types.
     e. From the third drop-down list, select the implementation type that is necessary for your application. If your application does not require that connections support two-phase commit transactions, choose **Connection Pool Data Source**. Choose **XA Data Source**, however, if your application requires connections that support two-phase commit transactions. Applications that use this data source configuration have the benefit of container-managed transaction recovery.

        After you select an implementation type, the wizard fills the name and the description fields for your JDBC provider. You can type different values for these fields; they exist for administrative purposes only.
     f. Click **Next** after you have defined your database type, provider type, and implementation type. Now you see the wizard panel Enter database class path information.

g. In the class path field, type the full path location of the database JDBC driver class files. Your class path information becomes the value of the WebSphere environment variable that is displayed on this panel, in the form of ${*DATABASE*_JDBC_DRIVER_PATH}. The application server uses the variable to define your JDBC provider; this practice eliminates the need to specify static JDBC class paths for individual applications. Remember that if you do not provide the full, correct JDBC driver class path for the variable, your data source ultimately fails. If the field already displays a fully qualified class path, you can accept that variable definition by completing the rest of this wizard panel and clicking **Next**.

h. Use the **Native library path** field to specify additional class files that your JDBC driver might require to function properly on your application server platform. Type the full directory path name of these class files.

i. Click **Next**. You now see the **Enter database specific properties for the data source** panel.

9. Complete all of the fields on the **Enter database specific properties for the data source** panel.
   - Click **Use this data source in container managed persistence (CMP)** if container managed persistence (CMP) enterprise beans must access this data source.
   - Any other property fields that are displayed on this wizard panel are specific to your database type. See the topic, Data source minimum required settings, by vendor, for information on these property settings. The article addresses both current and deprecated JDBC providers that are pre-defined in the application server.

     User-defined data sources: This wizard panel does not display additional property fields for data sources that correspond with your user-defined JDBC providers. However, from the JDBC driver class files that you installed, the application server can generally extract the necessary data source property names. The application server defines them as data source custom properties, displays them on a custom properties console panel, and assigns them default values. Consult your database documentation about setting these properties and any other requirements for your user-defined data source. After you create the data source, navigate to the corresponding custom properties collection panel in the administrative console by clicking **Data sources** > ***data_source*** > **Custom properties**. Review the property default values and modify them if necessary.

10. Optional: Configure the security aliases for the data source. You can select none for any of the authentication methods, or choose one of the following types:
    - **Component-managed authentication alias** - specifies an authentication alias to use when the component resource reference res-auth value is Application. To define a new alias, navigate to **Related Items** > **J2EE Connector Architecture (J2C) authentication data entries**. A component-managed alias represents a combination of ID and password that is specified in an application for data source authentication. Therefore, the alias that you set on the data source must be identical to the alias in the application code.

      a. Use the drop-down list to select an existing component-managed authentication alias.

      b. To create a new alias, click the links that are provided. This action closes the data source wizard and triggers the administrative console to display the J2C authentication data panel. Click **New** to define a new alias. Click **OK** to save your settings and view the new alias on the J2C authentication data panel. Restart the data source wizard by navigating back to the data source collection panel, selecting the appropriate scope, and clicking **New**.

      For more information on Java 2 Connector (J2C) security, see the topic on managing Java 2 Connector Architecture authentication data entries.
    - **Mapping-configuration alias** - is used only in the absence of a login configuration on the component resource reference. The specification of a login configuration and the associated properties on the component resource reference is the preferred way to define the authentication

strategy when the res-auth value is set to Container. If you specify the DefaultPrincipalMapping login configuration, the associated property is a JAAS - J2C authentication data entry alias.

- **Container-managed authentication alias** - is used only in the absence of a login configuration on the component resource reference. The specification of a login configuration and the associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is set to `Container`.

Note: If you have defined security domains in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. Security domains support isolating authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that are able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them.

11. Click **Next** to view the **Summary** panel, and review any information for the data source. If any information is not correct, you can click **Previous** to go back and correct it.
12. Click **Finish** to save the configuration and exit the wizard. You now see the **Data sources** panel, which displays your new configuration in a table along with other data sources that are configured for the same scope.

What to do next

You can override the default values for some data source properties. You can also configure additional properties that your database vendor might require or offers as options. Consult your database documentation about these settings. The following topics in this information center inform you of how to use the administrative console to assign the property values:

- “Connection pool settings” on page 205
- Tuning connection pools
- “WebSphere Application Server data source properties” on page 289
- “Java EE resource provider or connection factory custom properties collection” on page 296

Disabling statement pooling:

Disable statement pooling when performing some Data Definition Language (DDL) operations, which might not be compatible with statement pooling. DDL operations, such as dropping and recreating tables, are not compatible with statement pooling when using the IBM Informix database. DDL operations invalidate the pooled statements or cause them to produce unexpected results.

About this task

Complete the following steps to use the administrative console to disable statement pooling. Statement pooling is disabled by specifying a value of zero for the statement cache size setting for your datasource.

Procedure

1. From the administrative console, select **Resources** > **JDBC** > **Data sources** > ***datasource_name***.
2. Under Additional properties, click **WebSphere Application Server data source properties**.
3. Enter a value of zero (0) in the Statement cache size field.
4. Click **OK**, and then click **Save**.

Data source collection:

Use this page to view configured data sources, which are the resources that provide connections to your relational database.

You can access this administrative console page in one of two ways:

- **Resources** > **JDBC** > **Data sources**.
- **Resources** > **JDBC** > **JDBC providers** > ***JDBC_provider*** > **Data sources**.

Version requirement: If you are using the Enterprise JavaBeans (EJB) 1.0 specification and the Java Servlet 2.2 specification, you must use the **Data sources (WebSphere Application Server Version 4)** console page.

Name:

Specifies the display name of this data source.

Click a data source name to edit the data source configuration settings.

Data type String

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name for this data source.

Data type String

Scope:

Specifies the scope of the JDBC provider that encapsulates the driver implementation classes to support this data source. Only applications that are installed within this scope can use this data source.

Provider:

Specifies the JDBC provider that encapsulates the appropriate classes.

Description:

Specifies a text description of the data source.

Data type String

Category:

Specifies a string that you can use to classify or group a data source.

Data type String

Data source settings:

Use this panel to edit the properties of a data source.

You can access this administrative console page in one of two ways:

- **Resources** > **JDBC** > **Data sources** > ***data_source***
- **Resources** > **JDBC** > **JDBC providers** > ***JDBC_provider*** > **Data sources** > ***data_source***

Note: If your application uses an Enterprise JavaBeans (EJB) 1.1 or a Java Servlet 2.2 module, use the **Data sources (WebSphere Application Server V4)** > ***data_source*** console page.

Test connection:

Activates the test connection service for validating application connections to the data source.

Before you click **Test connection**, set your data source properties and click **Apply**.

Scope:

Specifies the scope of the JDBC provider that supports this data source. Only applications that are installed within this scope can use this data source.

Provider:

Specifies the resource adapter that WebSphere Application Server uses for this connection factory.

Provider can be set only when you create a new connection factory. The list shows all the existing resource adapters that are defined at the relevant scope. Select one from the list if you want to use an existing resource adapter as Provider.

Name:

Specifies the display name for the data source.

Valid characters for this name include letters and numbers, but NOT most of the special characters. For example, you can set this field to `Test Data Source`. But any name starting with a period (•) or containing special characters (\ /,: ; " *? < > | = + & % ' ` @) is not a valid name.

Data type String

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name.

Distributed computing environments often employ naming and directory services to obtain shared components and resources. Naming and directory services associate names with locations, services, information, and resources.

Naming services provide name-to-object mappings. Directory services provide information about objects and the search tools required to locate those objects.

There are many naming and directory service implementations, and the interfaces to them vary. JNDI provides a common interface that is used to access the various naming and directory services.

For example, you can use the name`jdbc/markSection`.

If you leave this field blank a JNDI name is generated from the name of the data source. For example, a data source name, `markSection`, generates a JNDI name of `jdbc/markSection`.

After you set this value, save it, and restart the server, you can see this string when you run the memory dump name space tool.

Data type String

Use this data source in container-managed persistence (CMP):

Specifies if this data source is used for container-managed persistence of enterprise beans.

This option triggers creation of a CMP connection factory, which corresponds to this data source, for the relational resource adapter.

Data type	Boolean
Default	True (enabled)

Description:

Specifies a text description for the resource.

Data type	String

Category:

Specifies a category string you can use to classify or group the resource.

Data type	String

Data store helper class name:

Specifies the name of the DataStoreHelper implementation class that extends the capabilities of your selected JDBC driver implementation class to perform database-specific functions.

The application server provides a set of DataStoreHelper implementation classes for each of the JDBC provider drivers that it supports. These implementation classes are in the package com.ibm.websphere.rsadapter. For example, if your JDBC provider is DB2, then your default DataStoreHelper class is com.ibm.websphere.rsadapter.DB2DataStoreHelper. The administrative console page you are viewing, however, might make multiple DataStoreHelper class names available to you in a drop-down list; be sure to select the one required by your database configuration. Otherwise, your application might not work correctly. If you want to use a DataStoreHelper class that is not listed in the drop-down list, select **Specify a user-defined DataStoreHelper**, and type a fully qualified class name. See the information center for instructions on creating a custom DataStoreHelper class.

Data type	Drop-down list or string (if **user-defined DataStoreHelper** is selected)

Authentication alias for XA recovery:

This field is used to specify the authentication alias that you must use during XA recovery processing. If this alias name is changed after a server failure, the subsequent XA recovery processing uses the original setting that was in effect before the failure.

Select an alias from the list.

To define a new alias that is not displayed in the list:

1. Click **Apply**. Under Related Items, you now see a listing for Java Platform, Enterprise Edition (Java EE) Connector Architecture (J2C) authentication data entries.
2. Click **J2EE Connector Architecture (J2C) authentication data entries**.
3. Click **New**.
4. Define an alias.

5. Click **OK** and **Save**. The console now displays an alias collection page that lists all of your configured aliases. Above the table, this page also displays the name of your connection factory in the breadcrumb path.
6. Click the name of your J2C connection factory to return to the configuration panel for the connection factory that you are creating.
7. Select the new alias in the Container-managed authentication alias list.
8. Click **Apply**.

The database identity for the XA recovery authentication alias on a data source must have authorization to do XA recovery. Depending on the authorization schema for your installation, this level of authorization might be different from the level of authorization that the identity must use to access database tables for an application.

If the resource adapter does not support XA transactions, this field does not display. The default value for this field is derived from the selected alias for application authentication, if one is specified.

If you have defined multiple security domains and multiple authentication aliases in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. The Browse button is only accessible if at least one security domain is defined and assigned a scope which is applicable to the resource that is being edited. Additionally, that security domain must contain at least one Java Authentication and Authorization Service (JAAS) Java 2 Connector (J2C) Authentication alias.

Security domains allow you to isolate authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that are able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases that you cannot use do not display.

Data type Drop-down list

Component-managed authentication alias:

This alias is used for database authentication at run time.

If your database is not secured, setting database authentication is not required. This is not recommended for a production environment.

Note: If you have a database that does not support user ID and password, like Cloudscape, do not set the alias in the component-managed authentication alias or container-managed authentication alias fields. Otherwise, you see the warning message in the system log to indicate that the user and password are not valid properties. This message is only a warning message; the data source is still created successfully.

Select an alias from the list.

To define a new alias that is not displayed in the list:

1. Click **Apply**. Under Related Items, you now see a listing for Java Platform, Enterprise Edition (Java EE) Connector Architecture (J2C) authentication data entries.
2. Click **J2EE Connector Architecture (J2C) authentication data entries**.
3. Click **New**.
4. Define an alias.
5. Click **OK** and **Save**. The console now displays an alias collection page that lists all of your configured aliases. Above the table, this page also displays the name of your connection factory in the breadcrumb path.

6. Click the name of your J2C connection factory to return to the configuration panel for the connection factory that you are creating.
7. Select the new alias in the Container-managed authentication alias list.
8. Click **Apply**.

If you have defined multiple security domains and multiple authentication aliases in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. The Browse button is only accessible if at least one security domain is defined and assigned a scope which is applicable to the resource that is being edited. Additionally, that security domain must contain at least one JAAS J2C Authentication alias.

If you do not set an alias through the component-managed authentication or otherwise, and your database requires the user ID and password to get a connection, an exception occurs during run time.

Data type Drop-down list

Container-managed authentication alias:

Specifies authentication data, which is a JAAS - J2C authentication data entry, for container-managed signon to the resource. Depending on the value that is selected for the **Mapping-configuration alias** setting, you can disable this setting.

Select an alias from the list.

To define a new alias that is not displayed in the list:

1. Click **Apply**. Under Related Items, you now see a listing for J2EE Connector Architecture (J2C) authentication data entries.
2. Click **JAAS - J2C authentication data**.
3. Click **New**.
4. Define an alias.
5. Click **OK** and **Save**. The console now displays an alias collection page that lists all of your configured aliases. Above the table, this page also displays the name of your connection factory in the breadcrumb path.
6. Click the name of your J2C connection factory to return to the configuration panel for the connection factory that you are creating.
7. Select the new alias in the Container-managed authentication alias list.
8. Click **Apply**.

If you have defined multiple security domains and multiple authentication aliases in the application server, you can click **Browse...** to select an authentication alias for the resource that you are configuring. The Browse button is only accessible if at least one security domain is defined and assigned a scope which is applicable to the resource that is being edited. Additionally, that security domain must contain at least one JAAS J2C Authentication alias.

Security domains allow you to isolate authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that can access each authentication alias. The tree view is tailored for each resource, so domains and aliases that you cannot use do not display.

Data type Drop-down list

Mapping-configuration alias:

Specifies the authentication alias for the Java Authentication and Authorization Service (JAAS) mapping configuration that is used by this connection factory.

The **DefaultPrincipalMapping** JAAS configuration maps the authentication alias to the user ID and password. You can define and use other mapping configurations.

Data type Drop-down list

Common and required data source properties: These properties are specific to the data source that corresponds to your selected JDBC provider. They are either required by the data source, or are especially useful for the data source. You can find a complete list of the properties required for all supported JDBC providers in the information center.

WebSphere Application Server data source properties:

Use this page to set advanced data source properties in the application server. These properties activate and configure services that the application server applies to data sources to customize connections within an application server. These properties do not affect connections within the database.

To access this administrative console page complete one of the following paths:

- **Resources** > **JDBC** > **Data sources** > ***data_source*** > **WebSphere Application Server data source properties**
- **Resources** > **JDBC** > **JDBC providers** > ***JDBC_provider*** > **Data sources** > ***data_source*** > **WebSphere Application Server data source properties**
- **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Application scoped resources** > ***data_source*** > **WebSphere Application Server data source properties**.

Statement cache size:

Specifies the number of statements that can be cached per connection. The application server caches a statement after you close that statement.

The WebSphere Application Server data source optimizes the processing of prepared statements and callable statements by caching those statements that are not used in an active connection. Both statement types help maximize the performance of transactions between your application and data store.

- A prepared statement is a precompiled SQL statement that is stored in a PreparedStatement object. The application server uses this object to run the SQL statement multiple times, as required by your application run time, with values that are determined by the run time.
- A callable statement is an SQL statement that contains a call to a stored procedure, which is a series of precompiled statements that perform a task and return a result. The statement is stored in the CallableStatement object. The application server uses this object to run a stored procedure multiple times, as required by your application run time, with values that are determined by the run time.

If the statement cache is not large enough, useful entries are discarded to make room for new entries. To determine the highest value for your cache size to avoid any cache discards, add the number of uniquely prepared statements and callable statements, as determined by the SQL string, concurrency, and the scroll type, for each application that uses this data source on a particular server. This value is the maximum number of possible statements that can be cached on a given connection over the life of the server. Setting the cache size to this value means that you never have cache discards. In general, configure a larger cache for applications with a greater number of statements.

You can also use the Tivoli Performance Viewer to minimize cache discards. Use a standard workload that represents a typical number of incoming client requests, use a fixed number of iterations, and use a standard set of configuration settings.

Note: The higher the statement cache, the more system resources are delayed. Therefore, if you set the number too high, you might lack resources because your system cannot open multiple prepared statements.

If there is a particular statement that you do not want the application server to cache, configure the statement poolability hint to `false`. The application server does not cache a statement if the poolability hint is set to `false`. The application specifies the statement poolability hints at run time.

In test applications, tuning the statement cache improves throughput from 10% to 20%. However, because of potential resource limitations, this tuning process might not always be possible.

Data type	Integer
Default	Default values depend on the database. Typically, this value is 10. For Informix versions 7.3, 9.2, 9.3, and 9.4, without the respective latest fixes, the default value must be 0. A default value of 0 means that there is no cache statement.

Enable multithreaded access detection: If checked, the following warning message is entered in the WebSphere Application Server system out log if multiple threads attempt to concurrently use the same connection handle. You can use this property to debug connection problems if you think the problems might be caused by multiple threads trying to use the same connection handle. Having multiple threads concurrently using the same connection handle is a programming model violation.

```
J2CA0167W: An attempt to concurrently use the same connection handle by multiple threads has
been detected. The connection handle is: {0}.
```

Enable database reauthentication: Indicates that the exact match on connections retrieved out of the application server connection pool (the connection pool search criteria does not include a user name and password) cannot exist. Instead, the connection reauthentication is done in the doConnectionSetupPerTransaction() of the DataStoreHelper class. The application server does not provide a connection reauthentication implementation at run time. Therefore, when you check this box, you must extend the DataStoreHelper class to provide implementation of the doConnectionSetupPerTransaction() method where the reauthentication occurs. If you do not complete this process, the application server might return unusable connections. For more information, read the API documentation for the com.ibm.websphere.rsadapter.DataStoreHelper#doConnectionSetupPerTransaction method.

Connection reauthentication can help improve performance by reducing opening and closing connections, particularly for applications that frequently request connections with different user names and passwords.

Note: You cannot enable database reauthentication if you select TrustedConnectionMapping for the mapping configuration alias.

Enable JMS one-phase optimization support: When you check this option, the application server uses Java Message Service (JMS) to get optimized connections from this data source. This property prevents Java database connectivity (JDBC) applications from sharing connections with container-managed persistence (CMP) applications. This option is not available if the JDBC provider of the data source is an XA provider.

Manage cached handles: Specifies whether the container tracks cached handles, which are connection handles that an application component holds active across transaction and method boundaries. You can use this property to debug connection problems, but tracking handles can cause large performance issues during run time.

If the **Manage cached handles** property is selected in the administrative console, and you clear it, the field is no longer visible for resources that are at Version 7.0 or greater of the application server. This field is only displayed if the manageCachedHandles property is set to `true` in the resources.xml file. To make the field available, change the value for the manageCachedHandles entry from `false` to `true` in the resources.xml file, or enter the following Jython command from the wsadmin tool:

```
AdminConfig.modify(myDataSourceVariable, '[[manageCachedHandles "true"]]')
```

Note: For any resources that are running at Version 6.x of the application server, the **Manage cached handles** property is always visible. For example, if you have a node that is at Version 6.1, the entry in the resources.xml file does not affect how the field is displayed in the administrative console.

For a different method to debug problems, use the multi-thread and cross-component diagnostic alerts to detect violations in the Java Connector Architecture (JCA) programming model. To enable these alerts, select those options from **Servers** > **Application servers** > ***application_server*** > **Performance** > **Performance and Diagnostic Advisor Configuration** > **Performance and Diagnostic Advice configuration** panel. These alerts force the connection manager to manage cached handles, detect the connection conditions, and send alerts.

Note: For these alerts to be active, you must also select **Enable Performance and Diagnostic Advisor Framework (Runtime Performance Advisor)** from the **Servers** > **Application servers** > ***application_server*** > **Performance** > **Performance and Diagnostic Advisor Configuration** panel.

Log missing transaction context: Specifies whether the container issues an entry to the activity log when an application obtains a connection without a transaction context. These are exceptions to the Java Platform, Enterprise Edition (Java EE) programming model connection requirements.

Non-transactional data source: Specifies that the application server does not enlist the connections from this data source in global or local transactions. Applications must explicitly call setAutoCommit(false) on the connection if they want to start a local transaction on the connection, and they must commit or roll back the transaction that they started.

Note: Set this property to `true` in rare circumstances, except when a Java Persistence API (JPA) application requires both JTA and non-JTA data sources. The non-JTA data source requires this property to be set to `true`.

Use WebSphere Application Server exception checking model:

Specifies that the application server uses the error mapping facility that is defined in the data store helper to identify errors. The application server does not replace exceptions that are thrown by the JDBC driver with exceptions that are defined in the error map of the data store helper.

Use WebSphere Application Server exception mapping model:

Specifies that the application server uses the error mapping facility that is defined in the data store helper to identify errors, and the application server replaces the exceptions that are thrown by the JDBC driver with exceptions that are defined in the error map of the data store helper.

Note: This error detection model functions with JDBC Version 3.0 and earlier.

Validate new connections: Specifies whether the connection manager tests newly created connections to the database.

Number of retries: Specifies the number of times you want to retry making the initial connection to a database after the first pretest operation fails.

Retry interval: If you select **Validate new connections**, this option specifies the length of time, in seconds, that the application server waits before retrying to make a connection if the initial attempt fails.

Validate existing pooled connections: Specifies whether the connection manager tests the validity of pooled connections before returning them to applications.

Retry interval: If you select **Pretest existing pooled connections**, this option specifies the length of time, in seconds, to allot to the JDBC driver for validating a connection.

Validation by JDBC driver:

Specifies that the application server uses the JDBC driver to validate the connections. The JDBC provider must support JDBC 4.0 or greater to use this option. This option is available only if either **Validate new connections** or **Validate existing pooled connections** is selected.

gotcha: For an Oracle data source, **Validation by JDBC Driver** displays on the administrative console only after the validateNewConnectionTimeout property is added to the custom properties of WebSphere Application Server data source properties. The validateNewConnectionTimeout property is used for JDBC 4.0 driver validation and can be specified with the administrative console.

Timeout: Specifies the timeout in seconds for testing connections, either new or pooled by the application server, to the database. If the timeout expires before validating then the connection is considered unusable. If retries are configured, the full value of the timeout applies to each retry. A value of 0 indicates the JDBC driver does not impose a timeout on validation attempts.

Note: This option is only available for JDBC drivers that are JDBC 4.0 compliant.

Validation by SQL string (deprecated):

Specifies an SQL statement that the application server sends to the database to test the connection. Use a query that is likely to have low impact on performance. This option is available only if either **Validate new connections** or **Validate existing pooled connections** is selected.

Optimize for get/use/close connection pattern with heterogenous pooling:

Optimizes the data source for applications that use the get/use/close connection pattern. This optimization enables the connection pool for the data source to share connections that are in the same transaction. With this optimization pattern, you can share one connection during a transaction even if connections use different connection properties.

If you use the heterogeneous pooling feature, you must first extend the data source definition so that you can specify different custom properties or applications to override non-core properties for the data source. For more information about extending data sources, see the information on extending DB2 data source definitions at the application level.

Note: This field is only available for DB2 data sources.

Retry interval for client reroute: Specifies the amount of time, in seconds, between retries for automatic client reroute.

Note: This field is only available for DB2 data sources.

Maximum retries for client reroute: Specifies the maximum number of connection retries that are attempted by the automatic client reroute function if the primary connection to the server fails. The property is only used when **Retry interval for client reroute** is set.

Note: This field is only available for DB2 data sources.

Alternate server names: Specifies the list of alternate server name or names for the DB2 server. If more than one alternate server name is specified, the names must be separated by commas. For example:

```
host1,host2
```

Note: This field is only available for DB2 data sources.

Alternate port numbers: Specifies the list of alternate server port or ports for the DB2 server. If more than one alternate server port is specified, the ports must be separated by commas. For example:

```
5000,50001
```

Note: This field is only available for DB2 data sources.

Client reroute server list JNDI name: Specifies the JNDI name that is used to bind the DB2 client reroute server list into the JNDI name space. The DB2 database server uses this name to look up the alternate server name list when the alternate server information is not already in memory. This option is not supported for type 2 data sources.

Note: This field is only available for DB2 data sources.

Unbind client reroute list from JNDI: Used with test connection only. When set to `true`, the Client reroute server list JNDI name is unbound from the JNDI name space after a test connection is issued.

Note: This field is only available for DB2 data sources.

Data source (WebSphere Application Server V4) collection:

Use this page to view the settings of a WebSphere Application Server Version 4.0 data source.

These data sources use the WebSphere Application Server Version 4.0 Connection Manager architecture. All Enterprise JavaBeans (EJB) 1.x modules must use a WebSphere Application Server Version 4.0 data source.

You can access this administrative console page in one of two ways:

- **Resources** > **JDBC** > **Data sources(WebSphere Application Server V4)**.
- **Resources** > **JDBC** > **JDBC providers** > ***JDBC_provider*** > **Data sources(WebSphere Application Server V4)**.

Name:

Specifies a text identifier of the data source.

Data type String

JNDI Name:

Specifies the Java Naming and Directory Interface (JNDI) name of the data source.

Data type String

Scope:

Specifies the scope of the JDBC provider that encapsulates the driver implementation classes to support this data source. Only applications that are installed within this scope can use this data source.

Provider:

Specifies the JDBC provider that encapsulates the appropriate classes.

Description:

Specifies a text description of the data source.

Data type String

Category:

Specifies a text string that you can use to classify or group the data source.

Data type String

Data source (WebSphere Application Server Version 4) settings:

Use this page to create a Version 4.0 style data source. This data source uses the WebSphere Application Server Version 4.0 connection manager architecture. All Enterprise JavaBeans (EJB) 1.x modules must use a Version 4.0 data source.

You can access this administrative console page in one of two ways:

- **Resources** > **JDBC** > **Data sources(WebSphere Application Server V4)** > ***data_source***.
- **Resources** > **JDBC** > **JDBC providers** > ***JDBC_provider*** > **Data sources(WebSphere Application Server V4)** > ***data_source***.

Scope:

Specifies the scope of the JDBC provider that encapsulates the driver implementation classes to support this data source. Only applications that are installed within this scope can use this data source.

Provider:

Specifies the JDBC provider that WebSphere Application Server uses for this data source.

The list shows all of the existing JDBC providers that are defined at the relevant scope. Select one from the list if you want to use an existing JDBC provider as Provider.

Create new provider:

Provides the option of configuring a new JDBC provider for the new data source.

Create New Provider is displayed only when you create, rather than edit, a data source.

If you click **Create New Provider** the **Create a new JDBC provider** wizard launches. Complete all of the wizard panels and click **Finish**. The administrative console now displays the Data sources (WebSphere Application Server V4) configuration page again, where you see the new JDBC provider name in the Provider field.

Name:

Specifies the display name for the resource.

For example, you can set this field to `Test Data Source`.

Data type String

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name.

Distributed computing environments often employ naming and directory services to obtain shared components and resources. Naming and directory services associate names with locations, services, information, and resources.

Naming services provide name-to-object mappings. Directory services provide information about objects and the search tools required to locate those objects.

There are many naming and directory service implementations, and the interfaces to them vary. JNDI provides a common interface that is used to access the various naming and directory services.

For example, you can use the name `jdbc/markSection`.

If you leave this field blank a JNDI name is generated from the name of the data source. For example, a data source name of markSection generates a JNDI name of `jdbc/markSection`.

After you set this value, save it, and restart the server, you can see this string when you run the memory dump name space tool.

Data type String

Description:

Specifies a text description for the resource.

Data type String

Category:

Specifies a category string that you can use to classify or group the resource.

Data type String

Database name:

Specifies the name of the database of the data source to access.

For example, you can call the database `SAMPLE`.

Data type String

Default user ID:

Specifies the user name to use for connecting to the database.

For example, you can use the ID `db2admin`.

Data type String

Default password:

Specifies the password used for connecting to the database.

For example, you can use the password `db2admin`.

Data type String

Java EE resource provider or connection factory custom properties collection:

Use this page to view the custom properties of a Java Platform, Enterprise Edition (Java EE) resource provider or connection factory.

You can configure custom property collections for numerous resource types. According to the resource type with which a collection is associated, your ability to add, delete, and modify individual properties and settings varies. Begin the configuration process by clicking the **Required** field to sort those column values in descending order. All of the required (true) values are then sorted at the beginning of the page. Be sure to set all required properties.

Note: The following list displays three custom properties that are deprecated in WebSphere Application Server Version 6.10. The product now offers these properties as pre-configured options, which are the replacement properties in the following list. To avoid runtime error messages, permanently disable the original custom properties by deleting them from the table on this administrative console page.

- validateNewConnection -- replaced by **Pretest new connection**
- validateNewConnectionRetryCount -- replaced by **Number of retries**
- validateNewConnectionRetryInterval -- replaced by **Retry interval**

To set the replacement properties, click **Data sources** > ***data_source*** > **WebSphere Application Server data source properties** in the administrative console.

Name:

Specifies the property name.

Data type String

Value:

Specifies the property value.

Data type Variable; see “Custom property settings” on page 297 for more information.

Description:

Specifies text to describe any bounds or well-defined values for this property.

Data type String

Required:

Specifies whether this property is required for the resource provider.

Data type Boolean or check box

Custom property settings:

Use this page to specify the attributes of custom properties that might be required for resource providers and resource factories.

According to the resource type with which a property collection is associated, your ability to modify individual property settings varies. Therefore, consider the following descriptions as a general reference for custom property settings. (The administrative console page that you are using to configure your custom property might only allow you to modify a subset of the following settings.)

You can access this administrative console page through several different paths, depending on the type of resource to which the custom properties belong, for example: **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **Custom Properties**.

Note: The following list displays three custom properties that are deprecated in WebSphere Application Server Version 6.10. The product now offers these properties as pre-configured options, which are the replacement properties in the following list. To avoid runtime error messages, permanently disable the original custom properties by deleting them from the table on this administrative console page.

- validateNewConnection -- replaced by **Pretest new connection**
- validateNewConnectionRetryCount -- replaced by **Number of retries**
- validateNewConnectionRetryInterval -- replaced by **Retry interval**

To set the replacement properties, click **Data sources** > ***data_source*** > **WebSphere Application Server data source properties** in the administrative console.

Required:

Specifies that a value must be provided for this property of the resource, otherwise a warning occurs. If you are creating a new custom property, this field is not available.

Data type Check box

Name:

Specifies the name associated with this property, for example, *PortNumber*, *ConnectionURL*.

Data type String

Value:

Specifies the value associated with this property of the resource.

Data type Determined by the Type field. If the Type is `java.lang.String`, the text entered into the Value field is interpreted as a string. Similarly, if the Type is `java.lang.Integer`, the Value field is interpreted as numeric. This means that the text that is entered for the Value field must be able to be interpreted as indicated by the Type field, otherwise an error occurs. For example, attempting to create a property with a Type of `java.lang.Integer` and a Value of *someValue* results in an error. Only a numeric value can be properly interpreted for any of the available numeric types.

Description:

Specifies text to describe any bounds or well-defined values for this property.

Data type String

Type:

Specifies the fully qualified Java data type of this property.

There are specific types that are valid:
- java.lang.Boolean
- java.lang.String
- java.lang.Integer
- java.lang.Double
- java.lang.Byte
- java.lang.Short
- java.lang.Long
- java.lang.Float

Data type Drop-down list

Custom Properties (Version 4) collection:

Use this page to view properties for a WebSphere Application Server Version 4.0 data source.

You can access this administrative console page in one of two ways:
- **Resources** > **JDBC** > **Data Sources (WebSphere Application Server V4)** > *data_source* > **Custom Properties**
- **Resources** > **JDBC** > **JDBC Providers** > *JDBC_provider* > **Data Sources (WebSphere Application Server V4)** > *data_source* > **Custom Properties**

Name:

Specifies the name of the custom property

Data type String

Value:

Specifies the value of the custom property.

Data type Integer

Description:

Specifies text to describe any bounds or well-defined values for this property.

Data type String

Required:

Specifies properties that are required for this resource.

Data type String

Custom property (Version 4) settings:

Use this page to add properties for a WebSphere Application Server Version 4.0 data source.

To view this administrative console page, click **Resources** > **JDBC** > **JDBC Providers**> *JDBC_provider* > **Data Sources (WebSphere Application Server V4)** > *data_source* > **Custom Properties** > *custom_property*.

Required:

Specifies properties that are required for this resource.

Data type Check box

Name:

Specifies the name associated with this property (PortNumber, ConnectionURL, etc).

Data type String

Value:

Specifies the value associated with this property in this property set.

Data type Integer

Description:

Specifies text to describe any bounds or well-defined values for this property.

Data type String

Type:

Specifies the fully qualified Java type of this property (java.lang.Integer, java.lang.Byte).

Data type String

ResourceManagement command group for the AdminTask object

You can use the Jython or Jacl scripting languages to configure resource providers with the wsadmin tool. The commands and parameters in the ResourceManagement group can be used to define and display properties for resource providers.

The ResourceManagement command group for the AdminTask object includes the following commands:

- "setResourceProperty"
- "showResourceProperties" on page 301

setResourceProperty

Use the setResourceProperty command to set the value of a specified property defined on a resource provider such as JDBCProvider or a connection factory such as DataSource or JMSConnectionFactory. If the property with specified key is defined already, then this command overrides the value. If no property with a specified key is defined, this command will add the property with specified key and value.

Target object

The configuration object ID of a resource provider or a connection factory.

Required parameters

-propertyName
Specifies the name of the property. (String, required)

-propertyValue
Specifies the value of a property. (String, required)

Optional parameters

-propertyType
Specifies the type of the property. The default value is java.lang.String. (String, optional)

-propertyDescription
Specifies the description of the defined property. (String, optional)

Sample output

The command does not return output.

Examples

Batch mode example usage:

- Using Jacl:

```
$AdminTask setResourceProperty {-propertyName test.property -propertyValue testValue}
```

- Using Jython string:

```
AdminTask.setResourceProperty('[-propertyName test.property -propertyValue testValue]')
```

- Using Jython list:

```
AdminTask.setResourceProperty(['-propertyName', 'test.property', '-propertyValue', 'testValue'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask setResourceProperty {-interactive}
```

- Using Jython:

```
AdminTask.setResourceProperty('-interactive')
```

showResourceProperties

Use the showResourceProperties command to list all of the property values that are defined on a resource provider such as JDBC provider or a connection factory such as data source or JMS connection factory.

Target object

The configuration object ID of a resource provider or a connection factory.

Required parameters

None.

Optional parameters

-propertyName
Specifies the name of the property. If you specify the property name, the value of the specified property name is returned. If you do not specify the property name, all property values will be listed. Each element in the list is a property name value pair. (String, optional)

Sample output

The command returns the property values that are defined on the resource provider or the connection factory that you specified.

Examples

Batch mode example usage:

- Using Jacl:

```
$AdminTask showResourceProperties {-propertyName test.property}
```

- Using Jython string:

```
print AdminTask.showResourceProperties('[-propertyName test.property]')
```

- Using Jython list:

```
print AdminTask.showResourceProperties(['-propertyName', 'test.property'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask showResourceProperties {-interactive}
```

- Using Jython:

```
print AdminTask.showResourceProperties('-interactive')
```

Creating and configuring a JDBC provider and data source using the JMX API

If your application requires access to a relational database using the Java Database Connectivity (JDBC) API, you can create the necessary JDBC provider and data source objects using the Java Management Extensions (JMX) API exclusively. Alternatively, you can use the JMX API in combination with the wsadmin scripting tool.

About this task

bprac: If you are using the JMX API to create a data source, you should use one of the product-provided templates if one exists for your specific JDBC provider. The product provides a template for every supported JDBC provider. See the topic, Data source minimum required settings, by vendor, for a list of all of the supported JDBC providers. When you use the administrative console to select one of the provided templates, the administrative console prompts you for values for all of the required properties for that data source. See the topic, Configuring a JDBC provider and data source, for

more information about how to use the administrative console to create a data source. If you decide to use the wsadmin scripting tool to create a data source, see the topic, Data source minimum required settings, by vendor, for a list of the required settings.

Example

Using the JMX API to create the configuration objects necessary to access relational databases. This code sample demonstrates how to use the WebSphere Application Server MBeans to create and configure the objects which are required to access a relational database from an enterprise application. These objects include: a JDBC provider, either XA or non-XA capable, a data source, an authentication alias, and optionally a Container Managed Persistence (CMP) connection factory. Details of configuring these options can be found in the header comments of the following source code. The sample also demonstrates how to use the data source MBean to reload the configuration changes into the running server.

```
/**
 * COPYRIGHT LICENSE: This information contains sample code provided in
 * source code form. You may copy, modify, and distribute this sample
 * program in any form without payment to IBM for the purposes of
 * developing, using, marketing or distributing application programs
 * conforming to the application programming interface for the operating
 * platform for which the sample code is written. Notwithstanding anything
 * to the contrary, IBM PROVIDES THE SAMPLE SOURCE CODE ON AN "AS IS" BASIS
 * AND IBM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, BUT NOT
 * LIMITED TO, ANY IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY,
 * SATISFACTORY QUALITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND ANY
 * WARRANTY OR CONDITION OF NON-INFRINGEMENT. IBM SHALL NOT BE LIABLE FOR
 * ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES ARISING
 * OUT OF THE USE OR OPERATION OF THE SAMPLE SOURCE CODE. IBM HAS NO
 * OBLIGATION TO PROVIDE MAINTENANCE, SUPPORT, UPDATES, ENHANCEMENTS OR
 * MODIFICATIONS TO THE SAMPLE SOURCE CODE.
 *
 * © Copyright IBM Corp. 2011.
 * All Rights Reserved. Licensed Materials - Property of IBM.
 */

import java.util.Properties;
import java.util.Set;

import javax.management.Attribute;
import javax.management.AttributeList;
import javax.management.MalformedObjectNameException;
import javax.management.ObjectName;

import com.ibm.websphere.management.AdminClient;
import com.ibm.websphere.management.AdminClientFactory;
import com.ibm.websphere.management.Session;
import com.ibm.websphere.management.configservice.ConfigServiceHelper;
import com.ibm.websphere.management.configservice.SystemAttributes;
import com.ibm.websphere.management.exception.ConnectorException;

/**
 * This sample code demonstrates the use of the WebSphere JMX APIs to create
 * a DB2 jdbc provider and data source with an authentication alias.
 * This sample code may be configured to:
 * - Create either node or server scoped objects
 * - Create either an XA or non-XA jdbc provider
 * - Create a connection factory so the data source may be used by CMP beans,
 * otherwise the data source may be used by BMP beans, session beans, or
 * servlets
 * These are configured by setting the value of class constants
 * (see "Program configuration settings" below)
 * The jdbc provider, data source and CMP connection factory (if configured)
 * are created using predefined templates, which is a best practice.
 * Note: there are templates which will create both the jdbc provider and a
 * data source at once, this sample intentionally creates both objects
 * separately for thoroughness
 *
 * To compile and run this program, put the admin client bundle
 * was_install_root\runtimes\com.ibm.ws.admin.client_8.0.0.jar
 * on the classpath in addition to the classfile that was obtained when
 * this source file was built.
 *
 * Once compiled, this program is run in it's own JVM and
 * uses the WebSphere Admin Client API to interact with a WebSphere
 * Application Server either locally or remotely to create the
 * JDBC resources discussed above.
 */

public class JDBCResourcesWithJMX {

    // program configuration settings
```

```
// if true, an XA capable jdbc provider is created, otherwise non-XA
private static final boolean createXAJDBCProvider = true;
// if true, a CMP connection factory will be created for use by CMP beans,
// otherwise the data source created can be used with BMP beans, session
// beans, or servlets
private static final boolean createCMPConnectionFactory = true;
// if true, create node-scoped objects, otherwise create server
private static final boolean createNodeScopedCfgObjs = true;
// end program configuration settings

private static final String jdbcProviderName =
    "Sample DB2 JDBC Provider";
private static final String jdbcProviderTemplateNameXA =
    "DB2 Using IBM JCC Driver Provider Only (XA)";
private static final String jdbcProviderTemplateName =
    "DB2 Using IBM JCC Driver Provider Only";
// an alias for authentication data
private static final String authDataAlias = "db2admin";
private static final String authAliasDescription =
    "Sample authentication alias";
private static final String authAliasUserID = "db2admin"; // user ID
private static final String authAliasPassword = "db2admin"; // password
private static final String dataSourceProviderTemplateNameXA =
    "DB2 Using IBM JCC Driver - XA DataSource";
private static final String dataSourceHelperClassName =
    "com.ibm.websphere.rsadapter.DB2DataStoreHelper";
private static final String cmpConnFactTemplateName =
    "CMPConnectorFactory";
// display name for data source(DS)and connection factory(CF)
private static final String dataSourceName = "SampleDataSource";
private static final String dataSourceTemplateName =
    "DB2 Using IBM JCC Driver - DataSource";

@SuppressWarnings("unused")
private static final String dbName = "SamplesDB"; // the database name

public static void main(String[] args) {
    JDBCResourcesWithJMX cds = new JDBCResourcesWithJMX();

    try {
        cds.run(args);
    }
    catch (Exception ex) {
        System.out.println("Caught exception: " + ex);
        ex.printStackTrace();
    }
}

/**
 * Creates a jdbc provider, data source, auth alias and connection
 * factory, if so configured. Neither the data source or conn factory, if
 * created, is bound into namespace until the server is restarted or an
 * application using it is started
 */
private void run(String[] args) throws Exception {

    // Initialize the AdminClient.
    Properties adminProps = new Properties();
    adminProps.setProperty(AdminClient.CONNECTOR_TYPE,
        AdminClient.CONNECTOR_TYPE_SOAP);
    adminProps.setProperty(AdminClient.CONNECTOR_HOST, "localhost");
    adminProps.setProperty(AdminClient.CONNECTOR_PORT, "8880");
    AdminClient adminClient =
        AdminClientFactory.createAdminClient(adminProps);

    Session session = new Session();

    // use node scope to create config objects
    ObjectName scope = null;
    if (createNodeScopedCfgObjs) {
        scope = ConfigServiceHelper.createObjectName(null, "Node", null);
    }
    else { // unless server scope is configured
        scope = ConfigServiceHelper.createObjectName(null, "Server",
            "server1");
    }

    // retrieve the ConfigService MBean
    ObjectName configServiceMBean =
        retrieveJMXMBean(adminClient, "ConfigService");

    // invoke mbean queryConfigObjects operation
    // to find either a server or node
    ObjectName[] matches = (ObjectName[])adminClient.invoke(
        configServiceMBean,
        "queryConfigObjects",
        // parameters for operation
        new Object[] {session, null, scope, null},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
```

```
            "javax.management.ObjectName",
            "javax.management.ObjectName",
            "javax.management.QueryExp"});
    scope = matches[0]; // use the first server or node object found

    // create a jdbc provider at the specified scope
    ObjectName jdbcProv =
        createJDBCProvider(adminClient, session, scope,
        configServiceMBean);

    // create an auth alias for the data source
    createAuthAlias(adminClient, session, configServiceMBean);

    // Retrieve built-in WebSphere Relational Resource Adapter (RRA) at
    // same scope as above
    ObjectName rra =
        ConfigServiceHelper.createObjectName(null, "J2CResourceAdapter",
            null);

    // invoke mbean queryConfigObjects operation
    matches = (ObjectName[])adminClient.invoke(
        configServiceMBean,
        "queryConfigObjects",
        // parameters for operation
        new Object[] {session, scope, rra, null},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "javax.management.ObjectName",
            "javax.management.ObjectName",
            "javax.management.QueryExp"});
    rra = matches[0]; // use the first object found at this scope

    // create a data source using the jdbc provider
    ObjectName dataSource =
        createDataSource(adminClient, session, scope, jdbcProv, rra,
            configServiceMBean);

    // if configured to do so, create a Connection Factory on the
    // built-in WebSphere RRA for use by CMP beans
    if (createCMPConnectionFactory) {
        createConnectionFactory(adminClient, session, rra, dataSource,
            configServiceMBean);
    }

    // invoke mbean save operation to persist the changes
    // to the configuration repository, if platform is ND
    // will also need to perform a node sync
    adminClient.invoke(
        configServiceMBean,
        "save",
        // parameters for operation
        new Object[] {session, false},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "boolean"});

    System.out.println("Configuration changes saved");

    // reload data source MBean
    reload(adminClient, session);
}

/**
 * Use the ConfigService MBean to create an authentication alias
 */
private void createAuthAlias(AdminClient adminClient, Session session,
    ObjectName configServiceMBean) throws Exception {

    // Find the parent security object
    ObjectName security =
        ConfigServiceHelper.createObjectName(null, "Security", null);
    // invoke mbean queryConfigObjects operation
    Object result = adminClient.invoke(
        configServiceMBean,
        "queryConfigObjects",
        // parameters for operation
        new Object[] {session, null, security, null},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "javax.management.ObjectName",
            "javax.management.ObjectName",
            "javax.management.QueryExp"});
    security = ((ObjectName[])result)[0];

    // Prepare the attribute list
    AttributeList authAliasAttrs = new AttributeList();
    authAliasAttrs.add(new Attribute("alias", authDataAlias));
    authAliasAttrs.add(new Attribute("userId", authAliasUserID));
    authAliasAttrs.add(new Attribute("password", authAliasPassword));
    authAliasAttrs.add(
```

```
        new Attribute("description", authAliasDescription));

    // invoke jmx createConfigData operation
    result = adminClient.invoke(configServiceMBean, "createConfigData",
        // parameters for operation
        new Object[] {session, security, "authDataEntries",
        "JAASAuthData", authAliasAttrs},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "javax.management.ObjectName",
            "java.lang.String",
            "java.lang.String",
            "javax.management.AttributeList"});

    System.out.println("Created authorization alias: " + authDataAlias);
}

/*
 * Use the ConfigService MBean to create a CMP connection factory
 * on the built-in WebSphere RRA
 */
private void createConnectionFactory(AdminClient adminClient,
    Session session, ObjectName rra, ObjectName dataSource,
    ObjectName configServiceMBean)
    throws Exception {

    // Prepare the attribute list
    AttributeList cfAttrs = new AttributeList();
    cfAttrs.add(new Attribute("name", dataSourceName + "_CF"));
    cfAttrs .add(new Attribute("authMechanismPreference",
        "BASIC_PASSWORD"));
    cfAttrs.add(new Attribute("authDataAlias", authDataAlias));
    cfAttrs.add(new Attribute("cmpDatasource", dataSource));

    // invoke jmx queryTemplates operation
    Object result = adminClient.invoke(
        configServiceMBean,
        "queryTemplates",
        // parameters for operation
        new Object[] {session, "CMPConnectorFactory",},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "java.lang.String"});

    // find the template with the desired display name attribute
    ObjectName connFactTemplate = null;
    if (result != null) {
        ObjectName[] templates = (ObjectName[])result;
        for (ObjectName template: templates) {
            if (cmpConnFactTemplateName.equals(template.getKeyProperty(
                SystemAttributes._WEBSPHERE_CONFIG_DATA_DISPLAY_NAME))) {
                connFactTemplate = template;
           }
        }
    }

    // use the template found above to create the CMP connection factory
    // invoke jmx createConfigDataByTemplate operation
    adminClient.invoke(
        configServiceMBean,
        "createConfigDataByTemplate",
        // parameters for operation
        new Object[] {session, rra, "CMPConnectorFactory",
            cfAttrs, connFactTemplate},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "javax.management.ObjectName",
            "java.lang.String",
            "javax.management.AttributeList",
            "javax.management.ObjectName"
            });

    System.out.println("Created CMP Connection factory: " +
        dataSourceName + "_CF");
}

/**
 * Use the ConfigService MBean to create a data source at the
 * specified scope from one of the predefined templates
 */
private ObjectName createDataSource(AdminClient adminClient,
    Session session, ObjectName scope, ObjectName jdbcProv,
    ObjectName rra, ObjectName configServiceMBean)
    throws Exception {

    // the template name to use based on whether this example is
    // configured to use an XA data source or not
    String templateName = dataSourceProviderTemplateNameXA;
    if (!createXAJDBCProvider) {
        templateName = dataSourceTemplateName;
```

```
    }

    // the attribute DataSource.relationResourceAdapter is required
    // in addition to the attributes in the template
    AttributeList dsAttrs = new AttributeList();
    dsAttrs.add(new Attribute("name", dataSourceName));
    // override some other props in the template
    dsAttrs.add(new Attribute("description", dataSourceName));
    dsAttrs.add(new Attribute("jndiName", "jdbc/" + dataSourceName));
    dsAttrs.add(new Attribute("datasourceHelperClassname",
        dataSourceHelperClassName));
    // link to the built-in WebSphere RRA
    dsAttrs.add(new Attribute("relationalResourceAdapter", rra));
    dsAttrs.add(new Attribute("authDataAlias", authDataAlias));

    // invoke jmx queryTemplates operation
    Object result = adminClient.invoke(
        configServiceMBean,
        "queryTemplates",
        // parameters for operation
        new Object[] {session, "DataSource"},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "java.lang.String"});

    // find the template with the desired display name attribute
    ObjectName db2Template = null;
    if (result != null) {
        ObjectName[] templates = (ObjectName[])result;
        for (ObjectName template: templates) {
            if (templateName.equals(template.getKeyProperty(
                SystemAttributes._WEBSPHERE_CONFIG_DATA_DISPLAY_NAME))) {
                db2Template = template;
           }
        }
    }

    // use the template found above to create the data source
    // invoke jmx createConfigDataByTemplate operation
    ObjectName dataSource = (ObjectName)adminClient.invoke(
        configServiceMBean,
        "createConfigDataByTemplate",
        // parameters for operation
        new Object[] {session, jdbcProv, "DataSource",
            dsAttrs, db2Template},
        // signature of bean method to invoke
        new String[] {"com.ibm.websphere.management.Session",
            "javax.management.ObjectName",
            "java.lang.String",
            "javax.management.AttributeList",
            "javax.management.ObjectName"
            });

    System.out.println("Created data source: " + dataSourceName +
        " at " + (createNodeScopedCfgObjs ? "node" : "server") +
        " scope");
    return dataSource;
}

/**
 * Get the DataSourceCfgHelper MBean and call reload() on it
 * This causes the newly created configuration objects to
 * be available.
 */
private void reload(AdminClient adminClient, Session session)
    throws Exception {

    // retrieve the DataSourceCfgHelper MBean
    ObjectName mBean =
        retrieveJMXMBean(adminClient, "DataSourceCfgHelper");

    // call the reload operation
    Object result =
        adminClient.invoke(
            mBean,
            "reload",
            new Object[] {},
            new String[] {});

    if (result != null) {
        System.err.println(
            "DataSourceCfgHelper MBean reload operation failed: "
            + result);
    }
    else {
        System.out.println("Reloaded DataSourceCfgHelper MBean");
    }
}

/**
```

```
     * Use the ConfigService MBean to create a jdbc provider at the specified
     * scope from one of the predefined templates
     */
    private ObjectName createJDBCProvider(
        AdminClient adminClient, Session session, ObjectName scope,
        ObjectName configServiceMBean) throws Exception {

        // the template name to use based on whether this example is
        // configured to use an XA jdbc provider or not
        String templateName = jdbcProviderTemplateNameXA;
        if (!createXAJDBCProvider) {
            templateName = jdbcProviderTemplateName;
        }

        // invoke jmx queryTemplates operation
        Object result = adminClient.invoke(
            configServiceMBean,
            "queryTemplates",
            // parameters for operation
            new Object[] {session, "JDBCProvider"},
            // signature of bean method to invoke
            new String[] {"com.ibm.websphere.management.Session",
                "java.lang.String"});

        // find the template with the desired display name attribute
        ObjectName db2Template = null;
        if (result != null) {
            ObjectName[] templates = (ObjectName[])result;
            for (ObjectName template: templates) {
                if (templateName.equals(template.getKeyProperty(
                    SystemAttributes._WEBSPHERE_CONFIG_DATA_DISPLAY_NAME))) {
                    db2Template = template;
                }
            }
        }

        // the attribute JDBCProvider.name is required in addition
        // to the attributes in the template
        AttributeList provAttrs = new AttributeList();
        provAttrs.add(new Attribute("name", jdbcProviderName
            + (createXAJDBCProvider ? " (XA)" : "")));
        // override the description in the template
        provAttrs.add(new Attribute("description", jdbcProviderName
            + (createXAJDBCProvider ? " (XA)" : "")));

        // use the template found above to create the jdbc provider
        // invoke jmx createConfigDataByTemplate operation
        ObjectName jdbcProvider = (ObjectName)adminClient.invoke(
            configServiceMBean,
            "createConfigDataByTemplate",
            // parameters for operation
            new Object[] {session, scope, "JDBCProvider",
                provAttrs, db2Template},
            // signature of bean method to invoke
            new String[] {"com.ibm.websphere.management.Session",
                "javax.management.ObjectName",
                "java.lang.String",
                "javax.management.AttributeList",
                "javax.management.ObjectName"
                });

        System.out.println("Created JDBC provider: " + jdbcProviderName
            + (createXAJDBCProvider ? " (XA)" : "")  +
            " at " + (createNodeScopedCfgObjs ? "node" : "server") +
            " scope");
        return jdbcProvider;
    }

    // find the specified MBean
    @SuppressWarnings("unchecked")
    private ObjectName retrieveJMXMBean(AdminClient adminClient,
        String beanName)
        throws MalformedObjectNameException, ConnectorException {
        // retrieve the ConfigService MBean
        ObjectName mBean = null;
        ObjectName queryName =
            new ObjectName("WebSphere:type=" + beanName + ",*");
        Set names = adminClient.queryNames(queryName, null);
        if (!names.isEmpty()) {
            mBean = (ObjectName) names.iterator().next();
        }
        return mBean;
    }
}
```

Example: Creating a JDBC provider and data source using Java Management Extensions API and the wsadmin scripting tool:

The following sample code is a JACL (wsadmin - scripting tool) script used to create a data source.

Use this script to create only data sources for which the product does *not* provide a template. For every JDBC provider WebSphere Application Server supports, the product provides a corresponding data source template. See the topic, Creating configuration objects using the wsadmin tool, for instructions on how to use the `createUsingTemplate` command to establish these data sources. For a complete list of supported JDBC providers (and therefore a complete list of data sources that must be created using a template), refer to the topic, Data source minimum required settings, by vendor.

This script sets up the following sample JDBC objects:
- Creates a data source *fvtDS_1*
- Creates a 4.0 data source *fvtDS_3*
- Creates a container-managed persistence (CMP) data source linked to *fvtDS_1*

Attention: If you later modify the class path or native library path of the JDBC provider associated with your data source, you must restart every application server within the scope of that JDBC provider for the new configuration to work. Otherwise, you receive a data source failure message.

```
#AWE --  Set up XA DB2 data sources, both Version 4.0 and  Connector architecture (JCA)-compliant data sources


#UPDATE THESE VALUES:
#The classpath that will be used by your database driver
set driverClassPath "c:/sqllib/java/db2java.zip"

set server "server1"

set fvtbase "c:/wssb/fvtbase"

#Users and passwords..
set defaultUser1 "dbuser1"
set defaultPassword1 "dbpwd1"
set aliasName "alias1"

set databaseName1 "jtest1"
set databaseName2 "jtest2"
#END OF UPDATES

puts "Add an alias alias1"
set cell [$AdminControl getCell]
set sec [$AdminConfig getid /Cell:$cell/Security:/]

#-------------------------------------------------------------
# Create a JAASAuthData object for component-managed authentication
#-------------------------------------------------------------
puts "create JAASAuthData object for alias1"

set alias_attr    [list alias $aliasName]
set desc_attr     [list description "Alias 1"]
set userid_attr   [list userId $defaultUser1]
set password_attr [list password $defaultPassword1]
set attrs [list $alias_attr $desc_attr $userid_attr $password_attr]

set authdata [$AdminConfig  create JAASAuthData $sec $attrs]
$AdminConfig save

set dm [$AdminControl queryNames type=Server,name=dmgr,*]
$AdminControl invoke $dm restart "true true"

puts "Installing DB2 datasource for XA"

puts "Finding the old JDBCProvider.."
#Remove the old jdbc provider...
set jps [$AdminConfig list JDBCProvider]
```

```
foreach jp $jps {
 set jpname [lindex [lindex [$AdminConfig show $jp {name}] 0] 1]
 if {($jpname == "FVTProvider")} {
  puts "Removing old JDBC Provider"
  $AdminConfig remove $jp
  $AdminConfig save
 }
}


#Get the server name...
puts "Finding the server $server"
set servlist [$AdminConfig list Server]
set servsize [llength $servlist]
foreach srvr $servlist {
 set sname [lindex [lindex [$AdminConfig show $srvr {name}] 0] 1]
 if {($sname == $server)} {
                puts "Found server $srvr"
  set serv $srvr
 }
}

set desiredNodeName "myNode"
puts "Finding the Node"
set nodelist [$AdminConfig list Node]
foreach node $nodelist {
  set nodename [lindex [lindex [$AdminConfig show $node {name}] 0] 1]
  if {($nodename == $desiredNodeName)} {
    puts "node = $node"
    break
  }
}

puts "Finding the Resource Adapter"
set ralist [$AdminConfig list J2CResourceAdapter $node]
set ralistlen [llength $ralist]
foreach ra $ralist {
  set raname [lindex [lindex [$AdminConfig show $ra {name}] 0] 1]
  if {($raname == "WebSphere Relational Resource Adapter")} {
    set rsadapter $ra
  }
}
#Create an iSeries JDBC Provider for the data sources
puts "Creating the provider for com.ibm.db2.jdbc.app.UDBXADataSource"
set attrs1 [subst {{classpath $driverClassPath}
 {implementationClassName com.ibm.db2.jdbc.app.UDBXADataSource}{name "FVTProvider2"}
 {description "DB2 UDB for iSeries JDBC Provider"} {xa "true"}}]
set provider1 [$AdminConfig create JDBCProvider $serv $attrs1]
#Create the first data source
puts "Creating the datasource fvtDS_1"
set attrs2 [subst {{name fvtDS_1} {description "FVT DataSource 1"}}]
set ds1 [$AdminConfig create DataSource $provider1  $attrs2]

#Set the properties for the data source.
set propSet1 [$AdminConfig create J2EEResourcePropertySet $ds1 {}]

set attrs3 [subst {{name databaseName} {type java.lang.String} {value $databaseName1}}]
$AdminConfig create J2EEResourceProperty $propSet1 $attrs3

set attrs10 [subst {{jndiName jdbc/fvtDS_1} {statementCacheSize 10}
 {datasourceHelperClassname com.ibm.websphere.rsadapter.DB2DataStoreHelper}
 {relationalResourceAdapter {$rsadapter}} {authMechanismPreference "BASIC_PASSWORD"}
 {authDataAlias  $aliasName}}]
$AdminConfig modify $ds1 $attrs10
```

```
#Create the connection pool object
$AdminConfig create ConnectionPool $ds1 {{connectionTimeout 1000}
 {maxConnections 30} {minConnections 1} {agedTimeout 1000}
 {reapTime 2000} {unusedTimeout 3000} }


#Create the 4.0 data sources
puts "Creating the 4.0 datasource fvtDS_3"
set ds3 [$AdminConfig create WAS40DataSource $provider1 {{name fvtDS_3} {description "FVT 4.0 DataSource"}}]

#Set the properties on the data source
set propSet3 [$AdminConfig create J2EEResourcePropertySet $ds3 {}]

#These attributes should be the same as fvtDS_1
set attrs4 [subst {{name user} {type java.lang.String} {value $defaultUser1}}]
set attrs5 [subst {{name password} {type java.lang.String} {value $defaultPassword1}}]
$AdminConfig create J2EEResourceProperty $propSet3 $attrs3
$AdminConfig create J2EEResourceProperty $propSet3 $attrs4
$AdminConfig create J2EEResourceProperty $propSet3 $attrs5
set attrs10 [subst {{jndiName jdbc/fvtDS_3} {databaseName $databaseName1}}]
$AdminConfig modify $ds3 $attrs10

$AdminConfig create WAS40ConnectionPool $ds3 {{orphanTimeout 3000} {connectionTimeout 1000}
 {minimumPoolSize 1} {maximumPoolSize 10} {idleTimeout 2000}}

#Add a CMP connection factory for the JCA-compliant data source. This step is not necessary for
#Version 4 data sources, as they contain built-in CMP connection factories.
puts "Creating the CMP Connector Factory for fvtDS_1"
set attrs12 [subst {{name "FVT DS 1_CF"} {authMechanismPreference BASIC_PASSWORD}
 {cmpDatasource $ds1} {authDataAlias $aliasName}}]
set cf1 [$AdminConfig create CMPConnectorFactory $rsadapter $attrs12]

#Set the properties for the data source.
$AdminConfig create MappingModule $cf1 {{mappingConfigAlias "DefaultPrincipalMapping"} {authDataAlias "alias1"}}

$AdminConfig save
```

Using the DB2 Universal JDBC Driver to access DB2 for z/OS

The z/OS operating system requires that you configure the DB2 Universal JDBC Driver and your database to ensure interoperability. Within WebSphere Application Server, configure a Java Database Connectivity (JDBC) provider object and a data source object to implement the driver capabilities for your applications.

Before you begin

The available versions of the DB2 Universal JDBC Driver to connect with DB2 on z/OS are as follows:

- The DB2 Universal JDBC Driver in DB2 UDB for z/OS Version 8. This version supports both driver Types 2 and 4.
- The DB2 Universal JDBC Driver for DB2 UDB for OS/390® and z/OS Version 7, as documented in APAR PQ80841. This version supports both driver Types 2 and 4.
- The DB2 Universal JDBC Driver with the feature *z/OS Application Connectivity to DB2 for z/OS*, which provides Type 4 connectivity only. If you install this version of the driver, you must configure a DB2 Universal JDBC Driver provider (XA) to access remote DB2 databases.

Consult the DB2 service updates for available enhancements on the version that use.

Migration tip: If you are replacing the DB2 for 390 and z/OS Legacy JDBC driver with the DB2 Universal JDBC Driver, you can migrate your existing JDBC provider settings. See the topic Migrating from the JDBC/SQLJ Driver for OS/390 and z/OS to the DB2 Universal JDBC Driver in the Information Management Software for z/OS Solutions Information Center for more information.

Procedure

1. Install the driver class files and any necessary native files in an available HFS directory.

   Native files are class files that some versions of the DB2 Universal JDBC Driver require for running on the z/OS operating system.

2. Configure the driver and database for interoperability

   a. Bind the required DB2 packages

      Any application that executes SQL statements in DB2 for z/OS, the Universal JDBC driver must first bind with DB2 the packages that represent the SQL statements to be executed.

      The specific details of the bind utility and bind process are described by the readme file provided with the installed DB2 Universal JDBC Driver. Refer to this readme file for details on how to set up and perform the required binding.

      The utility requires the server name or IP address, the port number, and the database name (the database location on z/OS) for the target DB2. To get this information, issue a DB2 **-DISPLAY DDF** command on the target DB2 system. This command displays the IPADDR (IP address), the SQL DOMAIN (server name), the TCPPORT number, and the LOCATION (database name/location) for you to use as input to the utility.

      You must perform the bind process for each target DB2 accessed using the DB2 Universal JDBC Driver.

   b. Set up to handle in-doubt transactions

      Perform this setup once for each target DB2 for z/OS Version 7 location that is accessed using the DB2 Universal JDBC Driver Type 4 XA support.

      DB2 for z/OS Version 7 does not implement Java Platform, Enterprise Edition (Java EE) XA support, therefore, the Type 4 driver XA processing uses DB2 V7 two-phase commit protocol and a table in each location (database) to store a list of global transactions that are in doubt or finished but not committed.

      This table must be set up at each DB2 V7 location that is accessed. To do the set up, use the In-Doubt Utility, which is included as part of the installed DB2 Universal JDBC Driver. The utility creates the SYSIBM.INDOUBT Table that stores information about In-Doubt Global Transactions. This utility also binds the package T4XAIndbtPkg, which contains the SQL statements to insert and delete from the SYSIBM.INDOUBT Table. The T4XAIndbtPkg package is written with SQLJ.

      This installation process requires that the target DB2 subsystem is configured with DDF enabled for incoming TCP/IP connections.

      1) To enable DDF on the target DB2, issue the DB2 **-START DDF** command on that system.
      2) This utility requires the server name or IP address and the port number for the target DB2 V7. To obtain this information, issue a DB2 **-DISPLAY DDF** command on the target DB2 V7 system. The IPADDR (IP address), the SQL DOMAIN (server name), and the TCPPORT number that can be used as input to the utility are displayed.

      To find more detailed information about the In-Doubt utility, refer to the *DB2 Universal Database™ for z/OS Version 7 Application Programming Guide and Reference for Java™* publication. You can download it from the Library section of the DB2 Universal Database for z/OS Version 7 product information web pages. Within this publication, search for discussion about the utility under **`DB2T4XAIndoubtUtil`**, which is the official name of the In-Doubt utility.

      Note: The previously described setup for in-doubt transactions is ***not*** a requirement for DB2 FOR z/OS Version 8 servers because DB2 FOR z/OS Version 8 natively supports XA commands over DRDA® and manages the In-Doubt Global Transactions internally.

   c. Define a db2.jcc.propertiesFile

      A db2.jcc.propertiesFile for use by DB2 Universal JDBC Driver Type 2 processing under WebSphere Application Server for z/OS can be created and specified as input to the driver. This runtime properties file is for use in specifying various runtime options that the DB2 Universal JDBC Driver uses for Type 2 connectivity. These options are specified as properties in the form of

parameter=value. Refer to the README file packaged with the installed DB2 Universal JDBC Driver for a detailed description of each of the properties.

This file is not required; however, if it is not provided, universal driver default processing is performed.

Of specific interest is the db2.jcc.ssid property. This property specifies the DB2 subsystem identifier (not location name), to be used by the DB2 Universal JDBC Driver Type 2 processing as the local subsystem name to which it connects. If this property is not provided, the driver uses the subsystem identifier that it finds in the DSNHDECP load module. If the installation wants to use the DSNHDECP load module to specify the subsystem identifier, this load module must be included in a `steplib` data set in the servant region PROCs associated with each server that uses the DB2 identified by the subsystem ID. Refer to the readme file packaged with the universal driver for more information about using this load module. If that DSNHDECP load module does not accurately reflect the wanted subsystem, or if multiple subsystems might be using a generic DSNHDECP, the db2.jcc.ssid property must be specified.

Although the db2.jcc.propertiesFile is not required, if you choose to define the file, you must specify the fully qualified HFS file name as a JVM System property as follows:

- **db2.jcc.propertiesFile = <fully-qualified-hfs-filename>**

The driver-general properties are typically specific to a driver load, for example, server, rather than all servers using the JDBC provider, therefore, it is best to set this JVM property at the server level. To define the **db2.jcc.propertiesFile=** property to the server level using the WebSphere Application Server for z/OS administrative console:

1) Under the WebSphere Application Server for z/OS administrative console, go to **Servers** > **Application Servers**, then click the server to which you want to add the JVM property.
2) On the selected server page, expand **Java and Process Management** and click **Process Definition** > **Servant**.
3) On the Servant page, click **Additional Properties**, then click **Java Virtual Machine**.
4) On the Java Virtual Machine page, click **Additional Properties**, then click **Custom Properties**.
5) On the Custom Properties page, scroll down and click **New** to configure a new JVM property for the selected server. The name of the property is **db2.jcc.propertiesFile**. The value of the property is the fully qualified HFS file name that you created and initialized with the DB2 Universal JDBC Driver properties. The Type 2 driver uses these properties for the selected server
6) Click **Ok**.
7) Click **Save** to save the new JVM property.

3. Define a JDBC provider for the DB2 Universal JDBC Driver. The JDBC provider object encapsulates the driver classes for implementation in WebSphere Application Server.
   a. From the WebSphere Application Server for z/OS administrative console, click **Resources** > **JDBC** > **JDBC Providers**.
   b. Select the *scope* at which applications can use the JDBC provider. This scope becomes the scope of any data source that you associate with this provider. You can choose a cell, node, cluster, or server. For more information, see the topic, administrative console scope settings.
   c. Click **New**. This action causes the **Create a new JDBC Provider** wizard to launch.
   d. Use the first drop-down list to select DB2 for z/OS as your database type.
   e. Select the DB2 Universal JDBC Driver provider as your JDBC provider type in the second drop-down list.
   f. From the third drop-down list, select the implementation type that is necessary for your application.

      If your application does not require that connections support two-phase commit transactions, and you plan to use type 4 connectivity, choose **Connection Pool Data Source**. If you use the connection pool data source with type 2 connectivity, however, Application Server on z/OS uses RRS to process *both* one-phase and two-phase transactions.

Restriction: Do not select **Connection Pool Data Source** if your installation has the z/OS Application Connectivity to DB2 for z/OS feature defined to WebSphere Application Server for z/OS. Only the XA implementation of the DB2 Universal JDBC Driver supports this feature.

Choose **XA Data Source** if you plan to use driver type 4, and your application requires connections that support two-phase commit transactions. Use only driverType 4 connectivity for the XA data source.

After you select an implementation type, the wizard fills the name and the description fields for your JDBC provider. You can type different values for these fields; they exist for administrative purposes only.

g. Click **Next** after you have defined your database type, provider type, and implementation type. Now you see the wizard page Enter database class path information.

   Typically you do not need to change the class path that already populates the field. (That class path is the value of the WebSphere environment variable that is displayed on this page, in the form of ${*DATABASE*_JDBC_DRIVER_PATH}.) Most likely you also do not need to change the native library path or the data source implementation class name.

h. Click **Next** to see a summary of your JDBC provider settings.

i. Click **Finish** if you are satisfied with the entire JDBC provider configuration. The JDBC provider collection page displays, which shows your new JDBC provider in a table along with other providers that are configured for the same scope.

4. Define a data source. WebSphere Application Server uses the data source object to obtain database connections and manage those connections.

   a. From the WebSphere Application Server for z/OS administrative console, access the page for the data source version that your applications require. If you need support for two-phase transactions, use only a data source of the latest standard version. Version 4 data sources do not support connections that participate in two-phase transactions.

      Navigate to the appropriate page in one of two ways:

      - Click **Resources** > **JDBC** > **Data sources**, or **Data sources (WebSphere Application Server Version 4)**.
      - Click **Resources** > **JDBC** > **JDBC providers** > *JDBC_provider* > **Data sources**, or **Data sources (WebSphere Application Server Version 4)**.

   b. Select the *scope* at which applications can use the data source. You can choose a cell, node, cluster, or server. For more information, see the topic Administrative console scope settings.

      Version 4 only: From this point onward, the steps for creating data sources (WebSphere Application Server Version 4) differ from the steps for creating data sources of the latest standard version. To configure a Version 4 data source complete the following steps:

      - Click **New** to proceed to the console page for defining required properties.
      - On this properties page specify values for the fields that are grouped under the heading **Configuration**. Application Server requires these properties to implement your JDBC driver classes; see the topic, Data source minimum required settings, by vendor to learn about acceptable values.
      - Save your configuration by clicking **OK**. You are now finished with the primary data source configuration tasks.
      - Optional: Define additional properties that are supported by the DB2 Universal JDBC provider. Application Server calls them *custom properties*, and requires that you set them on the data source as well. Begin by clicking the Custom Properties link that is now displayed on the administrative console page. You can learn about optional data source properties in the *Application Programming Guide and Reference for Java* for your version of DB2 for z/OS.

c. Click **New**. This action causes the **Create a data source** wizard to launch and display the Enter basic data source information page. The first field is the scope field, which is read-only. This field displays your previous scope selection.

d. Type a data source name in the Data source name field. This name identifies the data source for administrative purposes only.

e. Type a Java Naming and Directory Interface (JNDI) name in the JNDI name field. WebSphere Application Server uses the JNDI name to bind application resource references to this data source. For more information about JNDI, see the topic, Naming.

f. Configure a component-managed alias to secure your data source if you plan to implement driverType 4 connectivity with the DB2 Universal JDBC Driver. If you plan to use driverType 2 connectivity, you do not have to set an alias. In this case the connection manager uses a default authentication alias, which is the user identity of a thread when that thread delivers a getConnection request.

   A component-managed alias consists of an ID and password that are specified in an application for data source authentication. Therefore, the alias that you set on the data source must be identical to the alias in the application code. For more information about Java 2 Connector (J2C) security, see the topic, Managing Java 2 Connector Architecture authentication data entries.

   To set a component-managed alias, either select an existing alias or create a new one.

   - Use the drop-down list to select an existing component-managed authentication alias.
   - To create an alias, click the **create a new one** link. This action closes the data source wizard and triggers the administrative console to display the J2C authentication data collection page. Click **New** to define a new alias. Click **OK** to save your settings and view the new alias on the J2C authentication data collection page. Restart the data source wizard by navigating back to the data source collection page, selecting the appropriate scope, and clicking **New**.

g. Click **Next** to see the wizard page Select JDBC provider.

h. Either select an existing JDBC provider, or create a provider.

   To select an existing JDBC provider:

   1) Click **Select an existing JDBC provider**.
   2) Select a JDBC driver from the drop-down list.
   3) Click **Next**. You now see the page entitled Enter database-specific properties for the data source.

   To create a JDBC provider:

   1) Click **Create new JDBC provider**.
   2) Click **Next** to see the Create JDBC provider page.
   3) Use the first drop-down list to select DB2 for z/OS as your database type.
   4) Select the DB2 Universal JDBC Driver provider as your JDBC provider type in the second drop-down list.
   5) From the third drop-down list, select the implementation type that is necessary for your application.

      If your application does not require that connections support two-phase commit transactions, choose **Connection Pool Data Source**. Both driverType 2 and driverType 4 connectivity implementations of the DB2 Universal JDBC Driver support connection pool data sources.

      Restriction: Do not select this provider if your installation has the z/OS Application Connectivity to DB2 for z/OS feature defined to WebSphere Application Server for z/OS. Only the XA implementation of the DB2 Universal JDBC Driver supports this feature.

      Choose **XA Data Source** if your application requires connections that support two-phase commit transactions. Applications that use this data source configuration have the benefit of container-managed transaction recovery. Use only driverType 4 connectivity for the XA implementation.

After you select an implementation type, the wizard fills the name and the description fields for your JDBC provider. You can type different values for these fields; they exist for administrative purposes only.

6) Click **Next** after you have defined your database type, provider type, and implementation type. Now you see the wizard page Enter database class path information.

   Typically you do not need to change the class path that already populates the field. (That class path is the value of the WebSphere environment variable that is displayed on this page, in the form of ${*DATABASE*_JDBC_DRIVER_PATH}.) Most likely you also do not need to change the native library path or the data source implementation class name.

7) Click **Next**. You now see the page entitled Enter database-specific properties for the data source.

i. Click **Use this data source in container managed persistence (CMP)** if container managed persistence (CMP) enterprise beans must access this data source.

j. Specify all the remaining properties, as they are required for implementation of the DB2 Universal JDBC Driver. These properties include:

   - Database name, which is the location name of the target database used when establishing connections with this data source
   - driverType, which is the JDBC connectivity type used by the data source
   - Server name, which is the TCP/IP address or host name for the Distributed Relational Database Architecture™ (DRDA) server.

     This property is required only if driverType is set to 4. This property is not used if driverType is set to 2.
   - Port number, which is the TCP/IP port number where the DRDA server resides.

     Provide a value for this property only if driverType is set to 4. Do not set this property if driverType is set to 2.

k. Click **Finish** to save the configuration and exit the wizard. The Data source collection page displays, which shows your new configuration in a table along with other data sources that are configured for the same scope.

What to do next

You can override the default values for some data source properties. Click your new data source link in the table to view the general configuration page for required data source properties. You can also define additional properties that are supported by the DB2 Universal JDBC Driver. Application Server requires that you set them as custom properties on the data source. Learn about optional data source properties in the *Application Programming Guide and Reference for Java* for your version of DB2 for z/OS.

Extended data source properties:

Use this page to set the extended data source properties for a DB2 database. You can use these properties to allow an application to extend the custom properties for a data source or override any non-core properties that already exist for that data source.

To access this administrative console page:

- For applications that do not use container-managed persistence click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Resource references**
- For applications that use container-managed persistence click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Provide default data source mapping for modules containing 2.x entity beans**.

For data sources that use the DB2 Universal JDBC driver or DB2 Using IBM JCC Driver, click **Extended properties...** in the **Target Resource JNDI Name** column.

gotcha: If you include multiple values for an extended data source property, you must enclose those values in quotation marks.

Name: Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property that has that name is used.

Do not start your property names with `was`, because this prefix is reserved for properties that are predefined in the application server.

Data type String

Value:

Specifies the property value.

Data type Variable

Configuring two resource reference files on the same data source:

You can configure two resource reference files on the same data source. This allows you to extend the custom properties for the data source to be extended to include two different schema names (currentSQLId on z/OS or currentSchema name in the custom properties) that can be used to exploit the capabilities of the application server.

About this task

When an EntityManager is created, the application server obtains a connection to the database. When you are using a pessimistic transaction, the EntityManager will retain that connection until the EntityManager is closed. When there are two EntityManagers that extend the data source definitions, the `openjpa.jdbc.TransactionIsolation` property might cause a problem with the transaction. This property can be found in the `persistence.xml` file in the following entry:

```
property name="openjpa.jdbc.TransactionIsolation" value="read-committed"
```

In order to satisfy this request, Java Persistence API (JPA) will obtain a connection and immediately call `setTransactionIsolation(READ_COMMITTED)`. When you have two EntityManagers share a single physical connection to the database, the first EntityManager creates a connection to the database and involves that connection in a transaction. When the second EntityManager creates a connection, it is not able to change the isolation level.

You can avoid this problem by creating two resource reference files in the same data source. You can create the resource references with Rational Application Developer or by editing the XML files. You will need to make changes to the `ejb-jar.xml`, `ibm-ejb-jar-bnd.xml`, `ibm-ejb-jar-ext.xml`, `persistence.xml` files.

Note: For IBM Optim PureQuery Runtime, if this is an XA data source you must define a new custom property on the data source where *property_name* = `downgradeHoldCursorsUnderXa` and boolean value = `true`.

See the following sections for information on how to accomplish this:

- Configure two resource reference files on the same data source using Rational Application Developer.
- Configure two resource reference files on the same data source by editing the XML files.

Procedure

- Configure two resource reference files on the same data source using Rational Application Developer.
  1. Edit the `ejb-jar.xml` file
     a. Create the deployment descriptor if it doesn't already exist:
        1) Go to the context menu of the Enterprise Java beans (EJB) project and select **Java EE** > **Generate Deployment Descriptor Stub**.
     b. Edit the deployment descriptor:
        1) Go to the project's META-INF directory and select the `ejb-jar.xml` file.
     c. Add the enterprise bean's element to the deployment descriptor if it doesn't already exist:
        1) In the left hand pane, select **EJB Project node**. Click **Add**.
        2) In the dialog, select **Enterprise Beans**. Click **OK**.
     d. Add the session bean to the deployment descriptor if it's not already there:
        1) In the left pane, select **Enterprise Beans**. Click **Add**.
        2) Select **Session Bean**. Click **OK**.
        3) Enter the name of the session bean in the dialog. Click **OK**.
        4) In the right-side pane, enter the `EJB Class`, the business local, and the business remote interfaces.
     e. Add the resource reference elements to the session bean:
        1) Select the session bean in the left pane and click **Add**.
        2) Select **Resource Reference**. Click **OK**.
        3) In the **Add Resource Reference** dialog, enter the name, type, authentication and sharing scope fields. Click **OK**.
        4) Repeat for the second resource reference.
     f. Save the editor
  2. Edit the `ibm-ejb-jar-bnd.xml` file
     a. Create the WebSphere EJB bindings descriptor if it doesn't already exist
        1) Go to the context menu of the Enterprise JavaBeans (EJB) project and select **Java EE** > **Generate WebSphere Bindings Deployment Descriptor**.
     b. Edit the bindings descriptor:
        1) In the project's META-INF directory, select the `ibm-ejb-jar-bnd.xml` file.
     c. Add a binding element for the session bean to the bindings descriptor:
        1) In the left pane, select **EJB Jar Bindings** node and click **Add**.
        2) In the dialog, select **Session** and click **OK**.
        3) In the right pane, enter the name of the session bean.
     d. Add the bindings for the resource references to the session bean:
        1) In the left pane, select the session bean and click **Add**.
        2) In the dialog, select **Resource Reference** and click **OK**.
        3) In the left pane, select the resource reference.
        4) In the right pane, enter the name and binding name for the reference
        5) Repeat for the second resource reference.
     e. Save the editor
  3. Edit the `ibm-ejb-jar-ext.xml` file.
     a. Create the WebSphere EJB extensions descriptor if it doesn't already exist:
        1) Go to the context menu of the Enterprise JavaBeans (EJB) project and select **Java EE** > **Generate WebSphere Extensions Deployment Descriptor**.
     b. Edit the extensions descriptor:

1) In the project's META-INF directory, select the `ibm-ejb-jar-ext.xml` file.

c. Add an extensions element for the session bean to the extensions descriptor:
   1) In the left pane, select the **EJB Jar Extensions** node and click **Add**.
   2) In the dialog, select **Session** and click **OK**.
   3) In the right pane, enter the name of the session bean.

d. Define the isolation level for the resource references:
   1) In the left pane, select the session bean and click **Add**.
   2) In the dialog, select **Resource Reference** and click **OK**.
   3) In the left pane, select the resource reference.
   4) In the right pane, enter the name and the isolation level.
   5) Repeat for the second resource reference.

e. Save the editor

4. Edit the `persistence.xml` file.
   a. Add JPA support to the EJB project, which will create the persistence.xml file:
      1) From the project's context menu, select **Properties**.
      2) In the left pane, select the **Project Facets** node.
      3) In the right pane, check the box beside **Java Persistence**.
      4) Click **OK**.
   b. Edit the `persistence.xml` file:
      1) In the project's META-INF directory, select the `persistence.xml` file
   c. The created `persistence.xml` file already contains a persistence unit definition. Edit this `persistence.xml` file
      1) In the left pane, select the **Persistence Unit** node. Set the name, JTA data source and exclude the unlisted classes fields.
   d. Create a new persistence unit definition in the file:
      1) In the left pane, select the **Persistence** node and click **Add** .
      2) In the dialog, select **Persistence Unit** and click **OK**.
      3) In the left pan, select the **Persistence Unit** node. Set the name, JTA data source and exclude the unlisted classes fields.
   e. Add classes to the persistence unit:
      1) In the left pane, select the persistence unit node and click **Add** .
      2) In the dialog, select **Class** and click **OK**.
      3) In the right pane, enter the name of the class.
      4) Repeat for each class.
   f. Add properties to the persistence unit
      1) If there is not already a **Properties** element in the file, select the persistence unit node in the left pane and click **Add**.
      2) In the dialog, select **Properties** and click **OK**.
      3) In the left pane, select the **Properties** node and click **Add**.
      4) In the dialog, select **Property** and click **OK**.
      5) In the right pane, enter the name and value for the property
      6) Repeat the previous three steps to add the additional properties.
   g. Save the editor

- Configure two resource reference files on the same data source by editing the XML files.
  1. Edit the `ejb-jar.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar id="ejb-jar_ID" metadata-complete="false" version="3.0" xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/ejb-jar_3_0.xsd">
    <enterprise-beans>
        <session>
            <ejb-name>NewOrderSessionFacadeBean</ejb-name>
            <business-local>neworder session.ejb3.NewOrderSessionFacade</business-local>
            <business-remote>neworder session.ejb3.NewOrderSessionFacadeRemote</business-remote>
            <ejb-class>neworder session.ejb3.NewOrderSessionFacadeBean</ejb-class>
            <session-type>Stateless</session-type>
   <resource-ref>
    <description></description>
    <res-ref-name>jdbc/ERWWDataSourceV5</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
   </resource-ref>
   <resource-ref>
    <description></description>
    <res-ref-name>jdbc/ERWWDataSourceV5_HP</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
   </resource-ref>
        </session>
     </enterprise-beans>
</ejb-jar>
```

2. Edit the `ibm-ejb-jar-bnd.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar-bnd xmlns="http://websphere.ibm.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-ejb-jar-bnd_1_0.xsd"
 version="1.0">
 <session name="NewOrderSessionFacadeBean" simple-binding-name="ejb/session/NewOrderSessionFacadeBean">
  <resource-ref name="jdbc/ERWWDataSourceV5" binding-name="jdbc/ERWWDataSourceV5"></resource-ref>
  <resource-ref name="jdbc/ERWWDataSourceV5_HP" binding-name="jdbc/ERWWDataSourceV5"></resource-ref>
 </session>
</ejb-jar-bnd>
```

3. Edit the `ibm-ejb-jar-ext.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<ejb-jar-ext xmlns="http://websphere.ibm.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-ejb-jar-ext_1_0.xsd"
 version="1.0" metadata-complete="true">
 <session name="NewOrderSessionFacadeBean">
  <resource-ref name="jdbc/ERWWDataSourceV5"
   isolation-level="TRANSACTION_READ_COMMITTED" />
  <resource-ref name="jdbc/ERWWDataSourceV5_HP"
   isolation-level="TRANSACTION_READ_COMMITTED" />
 </session>
</ejb-jar-ext>
```

4. Edit the `persistence.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 version="1.0" xsi:schemaLocation=
"http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd">
    <persistence-unit name="NewOrderSessionEJB3">
        <jta-data-source>java:comp/env/jdbc/ERWWDataSourceV5</jta-data-source>
        <class>warehouseejb3.WarehouseJPA</class>
        <class>districtejb3.DistrictJPA</class>
        <class>customerejb3.CustomerJPA</class>
        <class>stockejb3.StockJPA</class>
        <class>orderejb3.OrderJPA</class>
        <class>orderlineejb3.OrderlineJPA</class>
        <class>neworderejb3.NewOrderJPA</class>
        <exclude-unlisted-classes>true</exclude-unlisted-classes>
        <properties>
  <property name="openjpa.LockManager" value="pessimistic"/>
  <property name="openjpa.ReadLockLevel" value="read"/>
  <property name="openjpa.WriteLockLevel" value="write"/>
  <property name="openjpa.LockTimeout" value="30000"/>
     <property name="openjpa.FetchBatchSize" value="1" />
     <property name="openjpa.jdbc.TransactionIsolation" value="read-committed" />
     <property name="openjpa.Log" value="none"/>
     </properties>
    </persistence-unit>
    <persistence-unit name="ItemEJB3">
        <jta-data-source>java:comp/env/jdbc/ERWWDataSourceV5_HP</jta-data-source>
        <class>itemejb3.ItemJPA</class>
```

```
        <exclude-unlisted-classes>true</exclude-unlisted-classes>
        <properties>
  <property name="openjpa.LockManager" value="pessimistic"/>
  <property name="openjpa.ReadLockLevel" value="read"/>
  <property name="openjpa.WriteLockLevel" value="write"/>
  <property name="openjpa.LockTimeout" value="30000"/>
     <property name="openjpa.FetchBatchSize" value="1" />
     <property name="openjpa.jdbc.TransactionIsolation" value="read-committed" />
     <property name="openjpa.Log" value="none"/>
     </properties>
    </persistence-unit>
</persistence>
```

Configuring Oracle Real Application Cluster (RAC) with the application server

Oracle Real Application Cluster (RAC) is a "share-everything" database architecture in which two or more Oracle RAC nodes are clustered together and share the same storage. The RAC nodes are connected together with a high-speed interconnect that enables fast communication between the Oracle nodes. The nodes can exchange various categories of data block ownership information during startup, lock information, exchange transaction information and data, and so on.

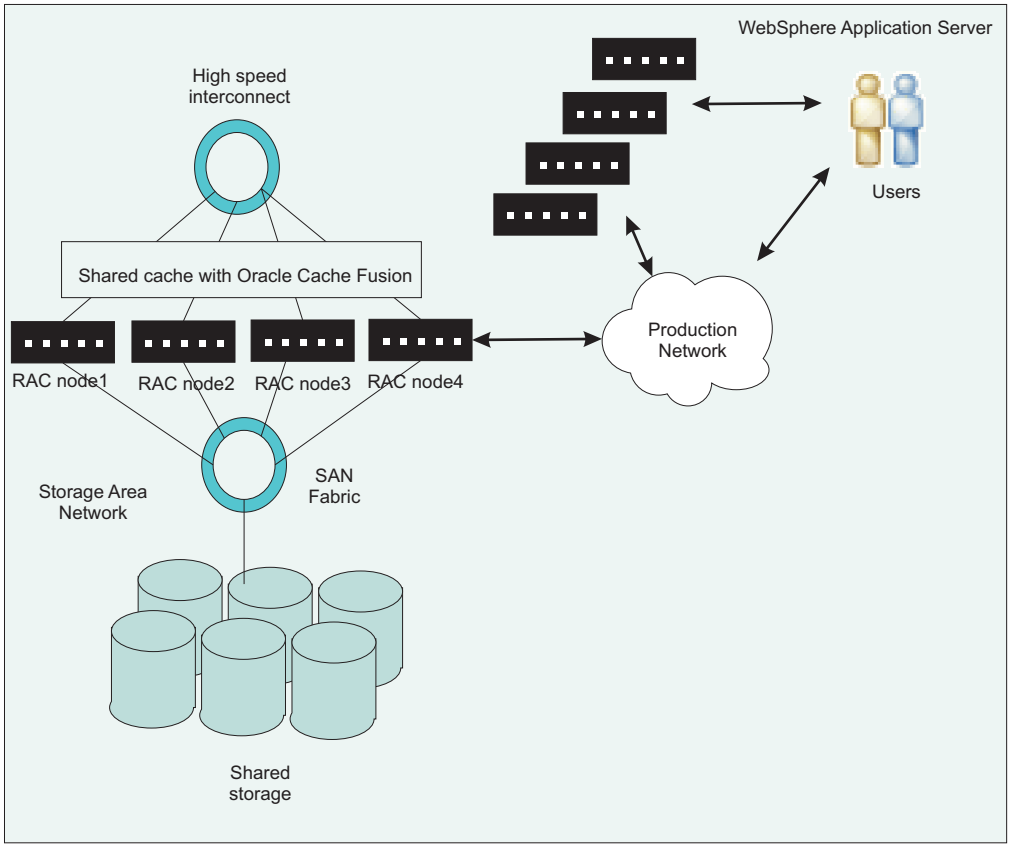
About this task

Using the Oracle JDBC driver, you can configure failover support, load balancing, or both, in an Oracle Real Application Clusters (RAC) environment. Oracle RAC is an option of an Oracle database that brings together two or more computers to form a clustered database that behaves as a single system. In a RAC database, Oracle processes that are running in separate nodes access the same data from a shared disk storage. First introduced in Oracle Version 9i, RAC provides both high availability and flexible scalability.

A typical Oracle RAC cluster consists of the following:

- **Cluster nodes** – 2 to *n* nodes or hosts, running the Oracle database server.
- **Network Interconnect** – a private network used for cluster communications and cache fusion. This is typically used for transferring database blocks between node instances.
- **Shared Storage** – used to hold the database system and data files. The shared storage is accessed by the cluster nodes.
- **Production network** – used by clients and application servers to access the database.

The following figure depicts a typical configuration for Oracle RAC:



Here are two of the many features that Oracle RAC provides:

- *Oracle Notification Service (ONS)* allows for Oracle RAC to communicate the status for the nodes, which are typically UP and DOWN events, to the Oracle JDBC driver and the driver's connection cache. To take advantage of ONS, you must configure the application server to use Oracle's connection caching instead of the application server's connection pooling feature. Read the topic Configuring Oracle connection caching in the application server for more information on this process.
- *Distributed Transaction Processing (DTP)* is a feature that was introduced in Oracle 10gR2. When this feature is enabled, Oracle will ensure that all in-flight prepared transactions that belong to a DTP service for failed RAC instances are pushed to disk. Then, Oracle will restart the DTP service on any of the RAC instances that are still operational.

For more information on Oracle RAC and how it works with the application server, refer to *Building a high availability database environment using WebSphere middleware: Part 3: Handling two-phase commit in WebSphere Application Server using Oracle RAC* on the developerWorks® website.

Procedure

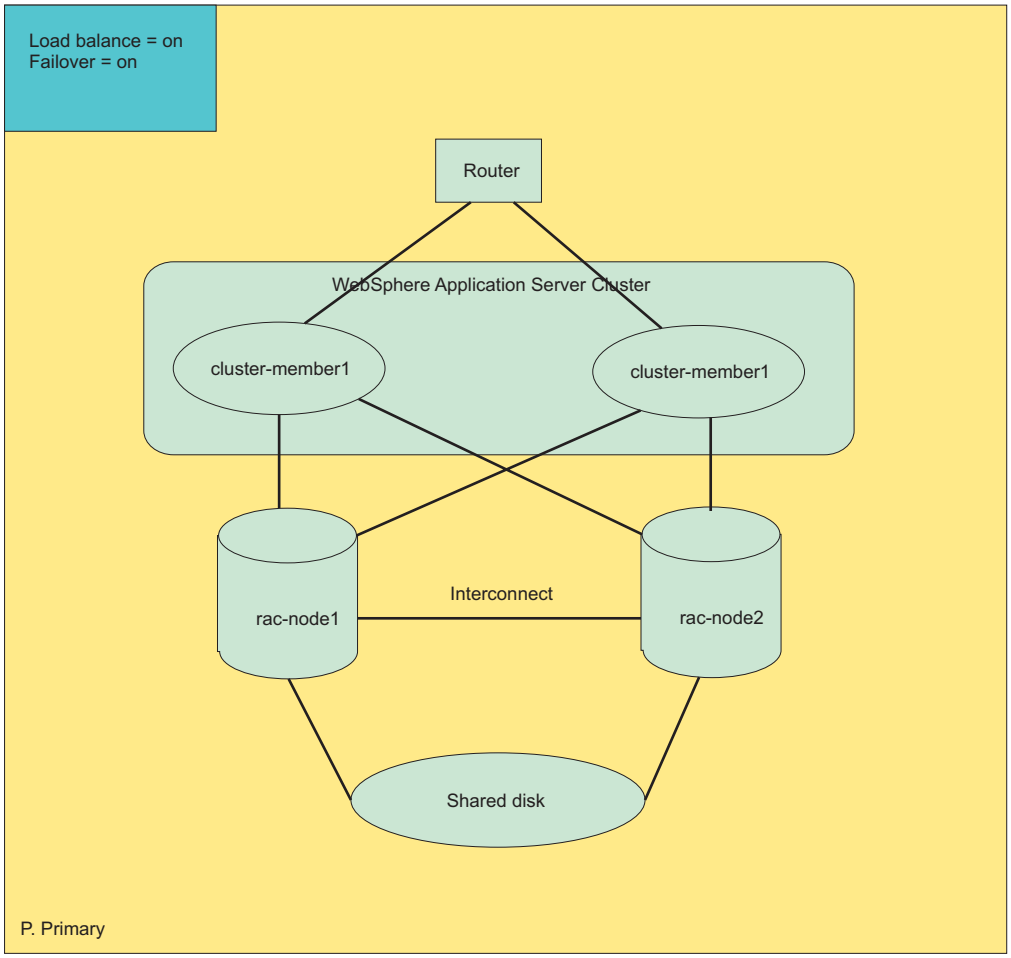
- "Configuring a simple RAC configuration in an application server cluster."
- "Configuring Oracle connection caching in the application server" on page 323.
- "Configuring two-phase commit distributed transactions with Oracle RAC" on page 324.

Configuring a simple RAC configuration in an application server cluster:

Oracle Real Application Cluster (RAC) is a "share-everything" database architecture that can provide high availability and load balancing. A typical configuration for an Oracle RAC contains two or more Oracle RAC nodes that are clustered together and share the same storage.

About this task

This figure depicts a typical RAC physical topology in a cluster environment for the application server, and both the failover and load balancing are enabled:



In the figure above, the application server cluster consists of two members: cluster-member1 and cluster-member2. The Oracle RAC physical configuration contains two nodes: rac-node1 and rac-node2. The RAC nodes can be located in the same physical machine with the cluster members, or they could be placed in entirely different machines. The actual placement does not impact the fundamental qualities of the services provided by RAC. To achieve both high availability and load-balancing, you can specify the Oracle data source URL for both cluster members in the application server with the required properties.

Procedure

1. Navigate to the Oracle data source. Click **Resources** > **JDBC** > **Data sources** > ***oracle_data_source***. If you don't already have an Oracle data source, create a new data source by clicking **New** and completing the wizard. For the URL, substitute the properties in the next step.
2. Set the URL for the Oracle database with the required configuration parameters.

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=
                  (ADDRESS=(PROTOCOL=TCP)(HOST=rac-node1)(port=1521))
                  (ADDRESS=(PROTOCOL=TCP)(HOST=rac-node2)(port=1521)))
                  (FAILOVER=on)(LOAD_BALANCE=on)
                  (CONNECT_DATA=(SERVER=DEDICATED)
                  (SERVICE_NAME=<service_name>)))
```

Note: Be aware of these configuration options:

- If you are not using Oracle services, then *service_name* will be the database name in the example. If you are using Oracle services, then *service_name* will be the name of the services.
- The example has FAILOVER and LOAD_BALANCE turned on. To turn one or both of these features off, change `on` to `off` in the above example.

3. Click **Apply** or **OK**.

Configuring Oracle connection caching in the application server:

You can elect to configure an Oracle data source to use the Oracle connection caching feature instead of using the application server connection pooling. Connection caching for Oracle databases is similar to connection pooling in the application server.

About this task

Currently, Oracle supports connection caching only with data sources that use the oracle.jdbc.pool.OracleDataSource implementation class, instead of the oracle.jdbc.pool.OracleConnectionPoolDataSource or oracle.jdbc.xa.client.OracleXADataSource classes. By default, the Oracle JDBC providers in the application server are configured to use the oracle.jdbc.pool.OracleConnectionPoolDataSource for non-XA data sources, or oracle.jdbc.xa.client.OracleXADataSource for XA data sources. To enable Oracle connection caching, you must configure and use a new JDBC provider in the application server that implements the oracle.jdbc.pool.OracleDataSource class.

Note: Oracle connection caching does not support XA.

Procedure

1. Create a data source and user-defined JDBC provider.
   a. Click **Resources** > **JDBC** > **Data sources**
   b. Select a server from the **Scope** drop-down list.
   c. Click **New**.
   d. Enter the name and JNDI name for the data source. Click **Next**.
   e. Create a JDBC provider. Select **Create new JDBC provider**, and click **Next**.
   f. Define the required properties for the JDBC provider. Use the following configuration settings:
      - **Database type:** User-defined
      - **Implementation class name:** oracle.jdbc.pool.OracleDataSource

      Click **Next**.
   g. Enter the class path for `ojdbc6.jar`, and click **Next**.
   h. For **Data store helper class name**, enter `com.ibm.websphere.rsadapter.Oracle11gDataStoreHelper`. Click **Next**.
   i. Define the security aliases for this data source, and click **Next**.
   j. Finish the wizard.
   k. Save the configuration changes.
2. Configure the data source that you created.
   a. Click the name of the data source. The configuration panel displays.
   b. Select **Custom properties**, and create or modify the properties for this data source. Enter or update the following custom properties:

Name	Value
disableWASConnectionPooling	true **gotcha:** You must also set the maximumPoolSize attribute to `0` on WebSphere Application Server connection pool settings to allow Oracle to control the pool boundaries.
connectionCachingEnabled	true
connectionCacheName	*your_cache_name*
removeExistingOracleConnectionPoolIfExists	true **Note:** The removeExistingOracleConnectionPoolIfExists property must be set to `true` so the application server removes any existing Oracle connection pools with an identical name. Otherwise, the Oracle data source fails the getConnection method if the pool name that is created has a name that is identical to an existing pool. For example, if you run a test connection, the test connection process creates an Oracle connection pool that prevents the application server from working properly at run time.
URL	*Oracle_URL*

Note: The order in which the custom properties are set is important. The setting order can be an issue because the application server passes the properties as a collection and the order is not guaranteed. If you encounter this issue, contact Oracle and reference Oracle bug #6638862.

3. Click **Apply** or **OK**.
4. Save the changes to the application server configuration.
5. Restart the application server.

Results

Oracle does not display a message if the pool creation fails, and a normal connection is returned instead. You can confirm that the Oracle connection pool is created by using the administrative console test connection function for the data source. First, turn on trace with the trace string, "RRA=all", for the server that runs your application. Then, issue a test connection. Issue a second test connection. Both test connections should work. Examine the trace log.

If the Oracle connection pool was created successfully, the trace shows that the second test connection detected that the Oracle connection cache exists because of the first test connection, and was successful in removing it so that it can be created again by the second test.

Configuring two-phase commit distributed transactions with Oracle RAC:

Real Application Cluster (RAC) configurations for Oracle 10g have an inherent issue with the transaction manager when Oracle attempts to recover two-phase commit distributed transactions that span over multiple Oracle RAC nodes. A problem can occur when one node fails, and Oracle opens up the other surviving nodes for business before the Oracle RAC completes the necessary recovery action for the node that has failed. The application server's ability to maintain transaction affinity provides you the ability to circumvent this issue.

About this task

Errors can occur when the recovery process attempts to commit or rollback a transaction branch through a RAC node that was previously active but later failed. The transaction manager would receive the following exception:

```
ORA- 24756: transaction does not exist
```

If this error is encountered, the Oracle database administrator might need to manually resolve the in-doubt transaction by forcing a rollback or commit process. If you do not desire a manual intervention, however, you might want to configure an automatic and transparent strategy for transaction recovery.

If the in-doubt transaction is not resolved, any subsequent transactions will receive the following exception:

```
ORA-01591 lock held by in-doubt distributed transaction
```

The result is that portions of the database will not be usable.

The key to a transparent recovery strategy is to eliminate the possibility of a global transaction spanning more than one transaction branch over multiple RAC nodes. A transaction branch corresponds to a database connection that is enlisted in a global transaction. If all connections in a global two-phase commit transaction originate from the same node, transaction recovery problems should not arise. Configure an Oracle RAC with the application server to prevent errors with two-phase transactions.

The application server maintains transaction affinity for incoming connections, and you can take advantage of this feature to configure automatic recovery for Oracle RAC with two-phase commit transactions. If you implement this configuration, all connections from a given application server will be received from the same Oracle node, and the connections will finish on that same node. This configuration will avoid situations in which transactions span multiple nodes, and you should not experience a recovery problem if one or more Oracle nodes go down.

Procedure

- You can elect to manually resolve the in-doubt transaction.
  1. Get the orphaned transaction ID. Issue the following command:

     ```
     sql > select state, local_tran_ID, Global_tran_Id from dba_2pc_pending where state = "prepared"
     ```
  2. Roll back all of the transaction IDs that are in the prepared phase.

     ```
     sql > rollback force '';
     ```
- Configure an automatic strategy for transaction recovery.
  1. Create an Oracle service that has only one primary node. Creating the service with one primary node will ensure that load balancing is disabled. You can also specify one or more alternate nodes with the -a parameter. Run this command to create the service:

     ```
     srvctl add service -d <database_name> -s <service_name> -r <primary nodes> -a <alternate_nodes>
     ```
  2. Enable Distributed Transaction Processing (DTP) on the Oracle service. DTP was first introduced in Oracle 10gR2. Each DTP service is a singleton service that is available on only one Oracle RAC instance. Run this command:

     ```
     execute dbms_service.modify_service (service_name => '<service_name>' , dtp => true);
     ```
  3. Configure each cluster member in the application server to use the Oracle DTP service.

Results

If you configured an automatic recovery strategy, the DTP service will start automatically on the preferred instance. However, if the database is restarted, the DTP service will not start automatically. You can start the DTP service using this command:

```
srvctl start service -d  -s
```

If a RAC node stops working, Oracle will not failover the DTP service until the Oracle RAC cleanup and recovery is complete. Even if the Oracle nodes come back up, the Oracle DTP service will not return to the freshly restarted RAC node. Instead, you will have to manually move the service to the restarted RAC node.

When you configure DTP on the Oracle service, you have transferred load balancing from the Oracle JDBC provider to the application server. The workload will be distributed by the application server instead of Oracle, which is why you created services that do not implement load balancing and only use one primary node. This configuration prevents situations in which transaction processes span multiple RAC nodes and alleviates recovery problems that can arise when one or more RAC nodes fail.

Configuring client reroute for applications that use DB2 databases

The client reroute feature enables you to configure your client applications for a DB2 universal database to recover from a communication loss, and the applications can continue to work with minimal interruption. Rerouting is central to the support of continuous operations, but rerouting is only possible when there is an alternate location that is identified to the client connection.

Before you begin

This task assumes the following:

- You have a DB2 data source defined in the application server. See the topic, Configuring a data source using the administrative console, for information about creating a data source.
- The DB2 data source to which your application connects is running one of the following:
  - DB2 for z/OS Version 9.1 or later
  - DB2 Database for Linux, UNIX, and Windows Version 9.5 or later
- You have implemented the DB2 database with a redundant setup or the ability to fail the DB2 server to a standby node.

About this task

Client reroute for DB2 allows you to provide an alternate server location, in case the connection to the database server fails. If you decide to use client reroute with the persistence option, the alternate server information persists across Java Virtual Machines (JVMs). In the event of an application server crash, the alternate server information is not lost when the application server is restored and attempts to connect to the database.

Without any configuration on the client side, a JDBC driver for DB2 supports the client reroute capability, if it is enabled, when the driver makes an initial connection to the DB2 server. When the JDBC driver connects to a DB2 server that has an alternate server configured, the primary server sends information about the alternate server to the JDBC driver. If the connection to the primary server fails, the JDBC driver is able to reroute connections to the alternate server. If the client process crashes, however, the alternate server information is lost, and the client needs to connect to the primary server again. If the client cannot make an initial connection to the primary server, the client has no knowledge of the alternate server and cannot reroute.

To overcome this problem, you can configure a DB2 data source in the application server with the **Alternate server name** and **Alternate port number** fields, or with the clientRerouteAlternateServerName and clientRerouteAlternatePortNumber data source custom properties, to support client reroute even on the initial connection attempt. If the JDBC driver is not able to connect to the primary DB2 server, the information that is necessary for a client reroute is already present, and the JDBC driver can reroute the connection to an alternate server.

Attention: The data source custom property, enableClientAffinitiesList, changes the semantics of the clientRerouteAlternateServerName and clientRerouteAlternatePortNumber properties.
To learn more about these properties, see the DB2 information center topic, Common IBM Data Server

Driver for JDBC and SQLJ properties for all supported database products. To learn more about client affinity, see the topic, .Configuring client affinity for applications that use DB2 databases.

Additionally, if you have configured a DB2 data source as a Type 4 JDBC driver, you can use the **Client reroute server list JNDI name** field, or the clientRerouteServerListJNDIName data source custom property, to enable persistence of the client reroute state. Typically, when a connection is rerouted and the JDBC driver has connected to the alternate DB2 server, the alternate server sends information about its own alternate server to the JDBC driver. The JDBC driver will then have the information that is required to reroute the connection again if the alternate DB2 server is not available. Effectively, the server that was originally the alternate server is now the primary server, and a new alternate server has been established. If you enable persistence for client reroute, this new state can be remembered. If the application server crashes and is restarted, the JDBC driver can connect to the DB2 server that was considered the primary server at the time of the crash. Without the persistence feature, the JDBC driver would have to start from the original server configuration and attempt to connect to the server that was originally considered the primary server.

You can use the automatic client rerouting feature within the following DB2 configurable environments:

- Enterprise Server Edition (ESE) with the data partitioning feature (DPF)
- Data Propagator (DPROPR)-style replication
- High availability cluster multiprocessor (HACMP™)
- High availability disaster recovery (HADR).

Procedure

1. In the administrative console, click **Resources** > **JDBC** > **Data sources** > ***data_source***.
2. Click **WebSphere Application Server data source properties**.
3. In the **DB2 automatic client reroute options** section, fill in the fields to enable client rerouting. Complete the following fields:

   Alternate server names
   Specifies the list of alternate server name or names for the DB2 server. If more than one alternate server name is specified, the names must be separated by commas. For example:

   ```
   host1,host2
   ```

   Alternate port numbers
   Specifies the list of alternate server port or ports for the DB2 server. If more than one alternate server port is specified, the ports must be separated by commas. For example:

   ```
   5000,50001
   ```

   Note: Ensure that an equal number of entries must be specified for both alternate ports and hosts. Otherwise, a warning is displayed and client reroute is not enabled.
4. Optional: Enable client reroute with the persistence option.
   a. Complete the field for **Client reroute server list JNDI name**. The field specifies the JNDI name that is used to bind the DB2 client reroute server list into the JNDI name space. The DB2 database server uses this name to look up the alternate server name list when the alternate server information is not already in memory.

      Note: Be aware of the following:
      - This option is not supported for Type 2 data sources. If you use a DB2 data source that is configured as a Type 2 JDBC driver, the JDBC driver uses a catalog to persist the client reroute information. If this property is configured with a Type 2 driver, the application server will issue a warning.
      - Use different JNDI names among different data sources. Otherwise, when you delete a data source, and the JNDI entry is removed from the name space, the other data sources that share the JNDI entry will be affected.

5. Configure the retry count and interval for the client reroute function. Complete these two fields:

   Retry interval for client reroute
   Specifies the amount of time, in seconds, between retries for automatic client reroute.

   Maximum retries for client reroute
   Specifies the maximum number of connection retries that are attempted by the automatic client reroute function if the primary connection to the server fails. The property is only used when **Retry interval for client reroute** is set.

   Attention: If you do not specify a value for these properties, DB2 failover processing (client rerouting) does not occur.
6. Click **OK** and save the changes.
7. Restart the application server.

What to do next

If you later want to remove the client reroute information that is bound in JNDI, you can do so by deleting the data source. You can also use the unbind feature with the test connection service to delete the JNDI binding for the client reroute function from the application server's JNDI name space without deleting the data source.

To delete the JNDI binding for client reroute:
1. Select **Unbind client reroute list from JNDI**.
2. Click **OK**.
3. Save the configuration.
4. Click **Test connection** for the data source.
5. Deselect **Unbind client reroute list from JNDI**.
6. Click **OK**.
7. Save the configuration.

Configuring client affinities for applications that use DB2 databases

The client affinities feature is an alternative to automatic client reroute when enabling your data source to use other servers when a connection fails. In this client-only method, the client determines the order that alternate servers run during failover. For more information about client affinities, see the topic, Client affinities for DB2 Database for Linux, UNIX, and Windows, in the DB2 information center.

Before you begin

This task assumes that:
- You have a DB2 data source defined in the application server. See the topic, Configuring a data source using the administrative console, for information about creating a data source.
- The DB2 data source to which your application connects is running one of the following databases:
  - DB2 for z/OS Version 9.1 or later
  - DB2 Database for Linux, UNIX, and Windows Version 9.5 or later
- You have implemented the DB2 database with a redundant setup or the ability to fail the DB2 server to a standby node.

About this task

In WebSphere Application Server, client affinities allows the DB2 data source on the client or application server to control the order of servers that are tried during initial connection processing.

For WebSphere Application Server, the data source custom property, enableClientAffinitiesList, is used to enable client affinities. If you want to use the administrative console to configure client affinities, use the DB2 automatic client reroute options section of the WebSphere Application Server data source properties panel to configure the following properties: Alternate server names, Alternate port numbers, Retry interval for client reroute, and Maximum retries for client reroute. You can also use the Custom properties panel to configure other client affinities properties as needed, including enableSeamlessFailover and affinityFailbackInterval.

The data source custom property, enableClientAffinitiesList, changes the semantics of the clientRerouteAlternateServerName and clientRerouteAlternatePortNumber properties.

Attention:

To learn more about these properties, see the DB2 information center topic, Configuration of client affinities for Java clients for DB2 Database for Linux, UNIX, and Windows connections.

To configure client affinities in the application server, complete the following steps:

Procedure

1. Configure a JDBC provider as usual with the JCC driver for DB2 in the class path.
2. Create a data source that uses the JDBC provider that was created in step 1. The server name and port information must be the name of the preferred primary server from your DB2 WLB environment.
3. After the data source is created, navigate to its primary panel in the administrative console by clicking **Resources** > **JDBC** > **Data sources** > ***data_source***.
4. Click **WebSphere Application Server data source properties** located under Additional Properties.
5. Scroll down to the section DB2 automatic client reroute options.
6. In the DB2 automatic client reroute options section, configure the Alternate server names, Alternate port numbers, Retry interval for client reroute, and Maximum retries for client reroute.
7. Click **OK**, and save the changes.
8. Navigate back to the data source primary panel, and click **Custom properties** located under Additional Properties.
9. On the Custom properties panel, configure other client affinities properties as needed, including enableClientAffinitiesList, enableSeamlessFailover and affinityFailbackInterval. Read about the configuration of client affinities for Java clients for DB2 Database for Linux, UNIX, and Windows connections, in the DB2 information center, for the recommended values for these properties.
10. Click **OK** and save the changes.
11. Restart the application server.

Verifying a data source connection

Many connection problems can be easily fixed by verifying configuration parameters. There are steps that you must complete to enable a successful connection.

About this task

If your connection is still not successful after completing these steps and reviewing the applicable information, check the SystemOut.log for warning or exception messages. Then use the technical support search function to find known problems. See the IBM Suggests section of this topic for a link to the IBM support and downloads page, which contains the search function.

Procedure

1. Create the authentication data alias. See the topic, Managing Java 2 Connector Architecture authentication data entries.
2. Create the JDBC provider. See the topic, Configuring JDBC providers using the administrative console.

3. Create a data source. See the topic, Configuring a data source using the administrative console
4. Save the data source.
5. Verify the connectivity if you created an authentication alias. To verify connectivity, restart the server for which you must verify connectivity.

   If you are using scripting or Java Management Extensions (JMX) to create the authentication alias, you can use the updateAuthDataCfg MBean. You can use this MBean method to refresh the authentication data in each server where it is needed. When you create a data source and the data source has not been used, you can use the updateAuthDataCfg MBean method. After a data source is used, its contents are instantiated in the server memory and generally cannot be changed.

   For more information about the updateAuthDataCfg MBean method, see the SecurityAdmin MBean documentation under Reference > Programming interfaces > Mbean interfaces.
6. Test the connection.

   You can test your connection from the data source collection view or the data source details view. Access either view in the administrative console, and then select a connection from the list. Click **Test Connection** on the connection. See the topic, Test the connection service.

Test connection service

WebSphere Application Server provides a test connection service for validating data source configurations. The testConnection operation instantiates the data source configuration, gets a connection, and then immediately closes the connection.

If you associate your data sources with WebSphere variables, see the topic, Creating, editing, and deleting WebSphere variables, to verify that you configure them correctly. A `variable cannot be found` exception results from attempted use of a data source that is invoked through an incorrectly defined variable.

Activating the test connection service

There are three ways to activate the test connection service: through the administrative console, the wsadmin tool, or a Java stand-alone program. Each process invokes the same methods on the same MBean.

Administrative console

WebSphere Application Server allows you to test a connection from the administrative console by simply pushing a button: the *Data source collection*, *Data source settings*, *Version 4 data source collection*, and *Version 4 data source settings* pages all have **Test Connection** buttons. After you define and save a data source, you can click this button to ensure that the parameters in the data source definition are correct. On the collection page, you can select several data sources and test them all at once. Note that there are certain conditions that must be met first. For more information, see the topic, Testing a connection with the administrative console.

Note: The following exception occurs when you click Test Connection to connect a Sybase data source from the administrative console.

```
Test connection failed for data source isagent on server server1 at node
svtaix24Node01 with the following exception: java.lang.Exception:
java.sql.SQLException: JZ006: Caught IOException: java.net.ConnectException: A
remote host refused an attempted connect operation.DSRA0010E: SQL State = JZ006,
Error Code = 0
```

This exception occurs when the Sybase data source port number is not matched to the port configured in Sybase server. The default port number is 5000. Check the port number of your Sybase server in the interfaces file under `/<sybase install directory>`.

WsAdmin tool

The wsadmin tool provides a scripting interface to a full range of WebSphere Application Server administration activities. Because the Test Connection functionality is implemented as a method on an MBean, and wsadmin can invoke MBean methods, wsadmin can be utilized to test connections to data sources. You have two options for testing a data source connection through wsadmin:

The *AdminControl* object of wsadmin has a testConnection operation that tests the configuration properties of a data source object. For information, see the topic, Testing a connection using wsadmin.

You can also test a connection by invoking the MBean operation. Use the example in the topic, Example: Testing data source connection using wsadmin, as a guide for this technique.

Java stand-alone program

Finally, you can test a connection by executing the `testConnection` method on the `DataSourceCfgHelper` MBean. This method allows you to pass the configuration ID of the configured data source. The Java program connects to a running Java Management Extensions (JMX) server to access the MBean. In a base installation of Application Server, you connect to the JMX server running in the application server, usually on port 8880.

The return value from this invocation is either 0, a positive number, or an exception. 0 indicates that the operation completed successfully, with no warnings. A positive number indicates that the operation completed successfully, with the number of warnings. An exception indicates that the test of the connection failed.

Example: Testing a connection using testConnection(ConfigID).

The following sample code creates a data source instance and an associated connection instance, and tests them to ensure database connectivity.

This program uses JMX to connect to a running server and invoke the *testConnection* method on the *DataSourceCfgHelper* MBean. The acronym *ND* in a comment line indicates that the following code applies to WebSphere Application Server WebSphere Application Server, Network Deployment. The word *Base* in a comment line indicates that the following code applies to WebSphere Application Server.

```
/**
 * Description
 * Resource adapter test program to make sure that the MBean interfaces work.
 * Following interfaces are tested
 *
 *  ---  testConnection()
 *
 *
 * We need following to run
 * C:\src>java -Djava.ext.dirs=C:\WebSphere\AppServer\lib;C:\WebSphere\AppServer\java\jre\lib\ext testDSGUI
 * must include jre for log.jar and mail.jar, else get class not found exception
 *
 *
 */

import java.util.Iterator;
import java.util.Locale;
import java.util.Properties;
import java.util.Set;

import javax.management.InstanceNotFoundException;
import javax.management.MBeanException;
import javax.management.MalformedObjectNameException;
import javax.management.ObjectName;
import javax.management.RuntimeMBeanException;
import javax.management.RuntimeOperationsException;
```

```
import com.ibm.websphere.management.AdminClient;
import com.ibm.websphere.management.AdminClientFactory;
import com.ibm.ws.rsadapter.exceptions.DataStoreAdapterException;

public class testDSGUI {

//Use port 8880 for a Base installation or port 8879 for ND installation
String port = "8880";
// String port = "8879";
String host = "localhost";
final static boolean verbose = true;

// eg a configuration ID for DataSource declared at the node level for Base
private static final String resURI = "cells/cat/nodes/cat:resources.xml#DataSource_1";

// eg a 4.0 DataSource declared at the node level for Base
//    private static final String resURI = "cells/cat/nodes/cat:resources.xml#WAS40DataSource_1";

// eg Apache Derby DataSource declared at the server level for Base
//private static final String resURI = "cells/cat/nodes/cat/servers/server1/resources.xml#DataSource_6";

// eg node level DataSource for ND
//private static final String resURI = "cells/catNetwork/nodes/cat:resources.xml#DataSource_1";

// eg server level DataSource for ND
//private static final String resURI = "cells/catNetwork/nodes/cat/servers/server1:resources.xml#DataSource_4";

// eg cell level DataSource for ND
//private static final String resURI = "cells/catNetwork:resources.xml#DataSource_1";

 public static void main(String[] args) {
  testDSGUI cds = new testDSGUI();
  cds.run(args);
 }

/**
 * This method tests the ResourceMbean.
 *
 * @param args
 * @exception Exception
 */
 public void run(String[] args) {

  try {

 System.out.println("Connecting to the application server.......");

       /*********************************************************************/
       /**    Initialize the AdminClient                                   */
       /*********************************************************************/
 Properties adminProps = new Properties();
 adminProps.setProperty(AdminClient.CONNECTOR_TYPE, AdminClient.CONNECTOR_TYPE_SOAP);
 adminProps.setProperty(AdminClient.CONNECTOR_HOST, host);
 adminProps.setProperty(AdminClient.CONNECTOR_PORT, port);
 AdminClient adminClient = null;
 try {
     adminClient = AdminClientFactory.createAdminClient(adminProps);
 } catch (com.ibm.websphere.management.exception.ConnectorException ce) {
       System.out.println("NLS: Cannot make a connection to the application server\n");
     ce.printStackTrace();
     System.exit(1);
 }

       /*********************************************************************/
       /**    Locate the Mbean                                             */
       /*********************************************************************/
 ObjectName handle = null;
```

```
try {
               // Send in a locator string
               // eg for a Base installation this is enough
               ObjectName queryName = new ObjectName("WebSphere:type=DataSourceCfgHelper,*");

               // for ND you need to specify which node/process you would like to test from
               // eg run in the server
//ND: ObjectName queryName = new OjectName
    ("WebSphere:cell=catNetwork,node=cat,process=server1,type=DataSourceCfgHelper,*");
               // eg run in the node agent
//ND: ObjectName queryName = new ObjectName
    ("WebSphere:cell=catNetwork,node=cat,process=nodeagent,type=DataSourceCfgHelper,*");
//ND: eg run in the   Manager
//ND: ObjectName queryName = new ObjectName
   ("WebSphere:cell=catNetwork,node=catManager,process=dmgr,type=DataSourceCfgHelper,*");
 Set s = adminClient.queryNames(queryName, null);
 Iterator iter = s.iterator();
 while (iter.hasNext()) {
               // use the first MBean that is found
  handle = (ObjectName) iter.next();
 System.out.println("Found this ->" + handle);
 }
 if (handle == null) {
  System.out.println("NLS: Did not find this MBean>>" + queryName);
  System.exit(1);
 }
} catch (MalformedObjectNameException mone) {
 System.out.println("Check the program variable queryName" + mone);
} catch (com.ibm.websphere.management.exception.ConnectorException ce) {
 System.out.println("Cannot connect to the application server" + ce);
}

        /*********************************************************************/
        /**           Build parameters to pass to Mbean                     */
        /*********************************************************************/
String[] signature = { "java.lang.String" };
Object[] params = { resURI };
Object result = null;

       if (verbose) {
 System.out.println("\nTesting connection to the database using " + handle);
}

try {
              /*********************************************************************/
              /**  Start to test the connection to the database                   */
              /*********************************************************************/
 result = adminClient.invoke(handle, "testConnection", params, signature);
} catch (MBeanException mbe) {
 // ****** all user exceptions come in here
 if (verbose) {
  Exception ex = mbe.getTargetException(); // this is the real exception from the Mbean
  System.out.println("\nNLS:Mbean Exception was received contains " + ex);
  ex.printStackTrace();
  System.exit(1);
 }
} catch (InstanceNotFoundException infe) {
 System.out.println("Cannot find " + infe);
} catch (RuntimeMBeanException rme) {
 Exception ex = rme.getTargetException();
 ex.printStackTrace(System.out);
 throw ex;
} catch (Exception ex) {
 System.out.println("\nUnexpected Exception occurred: " + ex);
 ex.printStackTrace();
}
```

```
        /*********************************************************************/
        /**  Process the result.  The result will be the number of warnings  */
        /**  issued.  A result of 0 indicates a successful connection with   */
        /**  no warnings.                                                     */
        /*********************************************************************/

 //A result of 0 indicates a successful connection with no warnings.
 System.out.println("Result= " + result);

  } catch (RuntimeOperationsException roe) {
 Exception ex = roe.getTargetException();
 ex.printStackTrace(System.out);
  } catch (Exception ex) {
 System.out.println("General exception occurred");
 ex.printStackTrace(System.out);
  }
 }
}
```

Tip: Ensure that you run the test connection service at the same level as an existing data source. For example, do not run the test connection service at the node level if your data source is configured on the server level. If the test connection service and the data source configuration do not exist on the same level, a failure to load exception might result. In this situation, source the db2profile script on the machine and ensure that the environment contains pointers to the DB2 native libraries. The db2profile script exists in the root directory of the DB2 user ID.

Testing a connection with the administrative console

After you have defined and saved a data source, you can click the **Test Connection** button to ensure that the parameters in the data source definition are correct.

Before you begin

About this task

You can select multiple data sources on the data source collection page and test them as a group. Be sure that the following conditions are met before using the Test Connection button:

Procedure

- If you are testing a connection using a WebSphere Application Server Version 4.0 type of data source, ensure that the *user* and *password* information is set.
- Designate variables appropriately.
  - WebSphere Application Server automatically sets environment variables for Java database connectivity (JDBC) driver class paths on the IBM i platform. Application Server names the variable according to your driver, either ${OS400_NATIVE_JDBC_DRIVER_PATH} or ${OS400_TOOLBOX_JDBC_DRIVER_PATH}. In the administrative console, you designate the complete path location of your driver as the value of the environment variable. See the “JDBC provider settings” on page 275 article for more information.
  - If you download the latest JTOpen version of the jt400.jar file, use one of two placement strategies to keep the value of your environment variable accurate. You can place the file in the same directory that you specify for the ${OS400_TOOLBOX_JDBC_DRIVER_PATH} variable. Alternatively, you can place the jt400.jar file in a different directory and change the value of ${OS400_TOOLBOX_JDBC_DRIVER_PATH} to this different path.
- Restart the application server after you define or edit WebSphere variables.
- Restart the application server after you create or edit an authentication alias for the data source.
- You can now test a connection to the data source. On the data source collection page in the administrative console, select the data source and click **Test Connection** .

A Test Connection operation can have three different outcomes, each resulting in a different message being displayed in the messages panel of the page on which you press the Test Connection button.

1. The test can complete successfully, meaning that a connection is successfully obtained to the database using the configured data source parameters. The resulting message states: Test Connection for data source *DataSourceName* on process *ProcessName* at node *NodeName* was successful.
2. The test can complete successfully with warnings. This means that while a connection is successfully obtained to the database, warnings were issued. The resulting message states: Test Connection for data source *DataSourceName* on process *ProcessName* at node *NodeName* was successful with warnings. View the JVM Logs for more details.

   The **View the JVM Logs** text is a hyperlink that takes you to the JVM Logs console screen for the process.
3. The test can fail. A connection to the database with the configured parameters is not obtained. The resulting message states: Test Connection failed for data source *DataSourceName* on process *ProcessName* at node *NodeName* with the following exception: *ExceptionText*. View the JVM Logs for more details.

   Again, the text for **View the JVM Logs** is a hyperlink to the appropriate logs screen.

Testing a connection using wsadmin

The AdminControl object of the wsadmin scripting tool has a testConnection operation that tests the configuration properties of a data source object.

Before you begin

About this task

The testConnection operation takes a data source *configuration ID* as an argument.

Note: This invocation cannot accept user IDs and passwords that must be defined in the database itself. This invocation can be used only for databases that do not require a user ID and password to make a connection (such as DB2 on a Windows machine), or for data sources that have a component-managed or container-managed authentication alias set on the data source object.

Procedure

1. Start the getid() method for your data source.
2. Set the value of the *configuration ID* to a variable.

   ```
   set myds [$AdminConfig getid /JDBCProvider:mydriver/DataSource:mydatasrc/]
   ```

   where *JDBCProvider:mydriver/DataSource:mydatasrc* is the data source you want to test. After you have the configuration ID of the data source, you can test the connection to the database.
3. Test the connection to the database. Use the following command:

   ```
   $AdminControl testConnection $myds
   ```

Configuring connection validation timeout

You can configure a timeout for connection validation by the Java Database Connectivity (JDBC) driver through a data source custom property in the data source configuration panels.

About this task

You can choose between validating connections with the JDBC driver or by having the application server run a SQL query. Select one or both of the following connection pretest attributes:

- Validate new connections
- Validate existing pooled connections

By default, connection validation is disabled. When you save the configuration for the data source, the administrative console supplies only the option that is selected. The administrative console will select validation by timeout or validation by a query, but if validation is not enabled then the application server will select neither option.

Procedure

1. Open the administrative console.
2. Go to the **WebSphere Application Server Data Source properties** panel for the data source.
   a. Select **Resources** > **JDBC** > **Data Sources** > ***data_source***
   b. Select **WebSphere Application Server Data Source properties**.
3. Go to the **Connection Validation Properties** section.
4. Select the type of connections that the application server will validate.
   - Select **Validate new connections**. This option specifies that the connection manager tests newly created connections to the database.
   - Select **Validate existing pooled connections**. This options specifies that the connection manager tests the validity of pooled connections before returning them to applications.
   - You can also select both options

     Note: You must make a selection here. If you do not select one or both of these options, you will not be able to select **Validation by JDBC Driver**. The **Validation by JDBC Driver** timeout feature is only available for JDBC providers that comply with the JDBC 4.0 specification.

     For an Oracle datasource, **Validation by JDBC Driver** appears on the administrative console only after the validateNewConnectionTimeout property is added to the custom properties of WebSphere Application Server datasource properties. The validateNewConnectionTimeout property is used for JDBC 4.0 driver validation and can be specified using administrative console.

   .
5. Click **Validation by JDBC Driver**. The application server issues a warning if **Validation by JDBC driver** is configured and the JDBC driver does not implement JDBC 4.0, or if the Connection.isValid method raises an error.

   Note: Connection validation by SQL query is deprecated. Use validation by JDBC Driver instead.
6. Enter the timeout value in the input box. The timeout value is in seconds.

   Note: If retries are configured, meaning the retry interval is not set to 0, for **Validate new connections** or **Validate existing pooled connections**, then the full value of the timeout applies to each retry. For each retry, the application server waits for the retry interval. Then the JDBC driver uses the full value of the timeout to validate the connection
7. Save the data source configuration.

What to do next

If you are modifying an existing data source, restart your server for this change to go into effect. If this is a new data source, restarting the server is not necessary.

Resource references

Use this page to designate how the resource references of application modules map to the actual resources that are configured for the application.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Resource references**.

You can also view this page during the **Map resource references to resources** step when you install an application.

- If your application uses any of the following resource types, you can set or reset their mapping configurations:
  - Default messaging JMS queues destinations
  - Default messaging JMS topic destinations
  - Data source
  - Generic JMS connection factory
  - Mail session
  - J2C connection factory
  - JMS queue connection factory for the JMS provider of WebSphere MQ
  - JMS queue destination for WebSphere MQ
  - JMS topic connection factory for WebSphere MQ
  - JMS topic destination for WebSphere MQ
  - Unified JMS connection factory for WebSphere MQ
  - URL configuration
- The page is composed of sections that correspond to each applicable resource type. Each section heading is the class name for the resource. If your application contains only one applicable resource type, you see only one section.
- Each section contains a table. Each table row depicts a resource reference within a specific module of your application.
- The rows contain the JNDI names of resource mapping targets for your references *only* if you bound them together during application assembly. You can modify those bindings on this administrative console page.
- To set your mappings:
  1. Select a row. If you want to apply the same mapping to multiple rows, complete the steps in the section, Set multiple JNDI names.
  2. Click **Browse** to view a new page listing of all resources that are available mapping targets for your application references.
  3. Select a resource and click **Apply**. The console displays the Resource references page again. The JNDI name of the selected resource mapping displays in the Target Resource JNDI Name field.
  4. Repeat the previous steps as necessary.
  5. If you are editing the resource references of an existing enterprise application, click **OK**. You now return to the general configuration page for your enterprise application. If you are installing the application and have completed the **Map resource references to resources** step, continue to the next step.
- **For data sources and connection factories:** Sections for these resource types contain an additional set of steps for modifying your security settings. Use the last column in the displayed table to view the authorization type for each resource configuration per application module. You can modify the corresponding authentication method only if the authorization type is container. Container-managed authorization indicates that the product performs signon to the resource rather than the enterprise bean code. The reconfiguring process differs slightly for each authentication method option:
  - When you want to assign no authentication method to a resource:
    1. Determine which resource configurations to designate with no authentication method.
    2. Select the appropriate table rows.
    3. Click **Modify Resource Authentication Method** and select **None** from the authentication method options that are displayed above the table.
    4. Click **Apply**.

– When you want to assign the WebSphere Application Server DefaultPrincipalMapping login configuration to a resource:
  1. You must apply this option to each resource individually if you want to designate different authentication data aliases. See the topic, J2EE connector security, for more information about the default mapping configuration.
  2. Select the appropriate table rows.
  3. Click **Modify Resource Authentication Method** and select **Use default method** from the list of authentication method options that are displayed above the table.
  4. Select an authentication data entry or alias from the list.
  5. Click **Apply**.
– When you want to assign a trusted context to a resource:
  1. You must have a data source that is running at least DB2 Version 9.1 for z/OS, and the data source must have trusted context enabled.
  2. You must have a data source server that is running at least DB2 Version 9.1 for z/OS, and the data source must have trusted context enabled.
  3. Select the appropriate table rows that have trusted context enabled.
  4. Click **Modify Resource Authentication Method** and select **Use trusted connections** from the authentication method options that are displayed above the table.
  5. Select an authentication alias from the list that matches an alias that is already defined in the DB2 data source. If you do not have an alias defined that is suitable, you must define a new alias.
  6. Click **Apply**.
  7. To edit the properties of the custom login configuration, click **Mapping Properties** in the table cell.
– When you want to assign a custom Java Authentication and Authorization Service (JAAS) login configuration to a resource:
  1. See the topic, J2EE connector security, for more information about custom JAAS login configurations.
  2. Select the appropriate table row.
  3. Click **Modify Resource Authentication Method** and select **Use custom login configuration** from the authentication method options that are displayed above the table.
  4. Select an application login configuration from the list.
  5. Click **Apply**.
  6. To edit the properties of the custom login configuration, click **Mapping Properties** in the table cell.

Set multiple JNDI names

Use this option to set the same JNDI name on multiple resources with one operation.

Click **Set multiple JNDI names** to display a menu of JNDI names. If you make a selection from this list, it is applied to the **Target Resource JNDI Name** field of all the selected rows of the table.

Modify Resource Authentication Method

Use this panel to toggle the display of a panel above the table rows.

This use of this panel is described in the **For data sources and connection factories** section.

Extended Properties

Use this panel to set additional properties on the selected resource.

Select a single table row and click **Extended Properties** to set additional properties on the selected resource. For more details on using this function, see the documentation on extending DB2 data source definitions at the application level.

Select
Select the check boxes of the rows that you want to edit.

Module
The name of a module in the application.

Bean
The name of an enterprise bean that is contained by the module.

URI
Specifies location of the module relative to the root of the application EAR file.

Resource Reference
The name of a resource reference that is used in the enterprise bean, if applicable, and is declared in the deployment descriptor of the application module.

Target Resource JNDI name
The Java Naming and Directory Interface (JNDI) name of the resource that is the mapping target of the resource reference.

Data type String

Login configuration
This column applies to data sources and connection factories only and refers to the authorization type and the authentication method for securing the resource.

Mapping-configuration alias
This panel allows you to select a mapping configuration alias for the resource that you are configuring. This panel is only available when security domains are defined. Security domains allow you to isolate mapping configuration aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that will be able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them. For example, a cell-scoped security domain will be hidden from the tree if all servers and clusters in the tree have defined their own security domain. If you are looking for an alias that is not visible in the tree, it is because the alias cannot be used by any servers that have visibility to this resource. In this case, you must define the alias at the global scope or in a different security domain that is visible to this resource.

To view this administrative console panel:

1. You must have a security domain defined in the application server.
2. Click one of the following paths in the administrative console:
   - **Resources** > **JDBC** > **Data sources** > ***data_source***. Click **Browse...** in the security section for **Component-managed authentication alias** or **Container-managed authentication alias**.
   - **Resources** > **JDBC** > **JDBC Providers** > ***jdbc_provider*** > **Data sources** > ***data_source***. Click **Browse...** in the security section for **Component-managed authentication alias** or **Container-managed authentication alias**.
   - **Resources** > **Resource Adapters** > **J2C connection factories** > ***j2c_connection_factory***. Click **Browse...** in the security section for **Component-managed authentication alias** or **Container-managed authentication alias**.

Note: Be careful when selecting an alias, because it is possible to select an alias that is only accessible by a subset of the servers that will use the resource. If you select a global alias, you are guaranteed that an alias by that name will be accessible to all users of the resource. If the alias has been overridden in a security domain, however, that alias will be used instead of the global one. The tree view includes icons to help you select the proper alias:

-  The alias is accessible by all servers that can access this resource.
-  There is at least one server that cannot access the alias. Check the tree view to see if this is OK for the application that will be using this resource.
-  The alias is defined in multiple places.

Select a J2C authentication alias

Use this panel to select a Java 2 Connector (J2C) authentication alias for the resource that you are configuring. This panel is available only when at least one security domain is defined and assigned a scope that is applicable to the resource that is being edited. Additionally, that security domain must contain at least one JAAS J2C Authentication alias. Security domains isolate J2C authentication aliases between servers.

The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that are able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them. For example, a cell-scoped security domain is hidden from the tree if all servers and clusters in the tree have defined their own security domain. If you are looking for an alias that is not visible in the tree, it is because the alias cannot be used by any servers that have visibility to this resource. In this case, you must define the alias at the global scope or in a different security domain that is visible to this resource.

To view this administrative console panel:

1. Read the preceding introduction for conditions under which this panel is accessible.
2. Click one of the following paths in the administrative console:
   - **Resources** > **JDBC** > **Data sources** > ***data_source***. Click **Browse...** in the security settings section for the applicable authentication classes.
   - **Resources** > **JDBC** > **JDBC providers** > ***jdbc_provider*** > **Data sources** > ***data_source***. Click **Browse...** in the security settings section for the applicable authentication classes.
   - **Resources** > **Resource Adapters** > **J2C connection factories** > ***j2c_connection_factory***. Click **Browse...** in the security settings section for the applicable authentication classes.
   - **Resources** > **Resource Adapters** > **J2C activation specifications** > ***j2c_activation_specification***. Click **Browse...** in the security settings section for the applicable authentication classes.
   - **Resources** > **JMS** > **Connection factories** > ***connection_factory***. Click **Browse...** in the security settings section for the applicable authentication classes.
   - **Resources** > **JMS** > **JMS providers** > ***jms_provider*** > **[Additional properties] Queue connection factories** > ***connection_factory***. Click **Browse...** in the security settings section for the applicable authentication classes.
   - **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application*** > **[Modules] Manage Modules** > ***module*** > **[Additional Properties] Resource Adapter** > **[Additional Properties] J2C connection factories** > ***connection_factory***. Click **Browse...** in the security settings section for the applicable authentication classes.
   - **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application*** > **[Modules] Manage Modules** > ***module*** > **[Additional Properties] Resource Adapter** > **[Additional Properties] J2C activation specifications** > ***activation_spec***. Click **Browse...** in the security settings section for the applicable authentication classes.

Note: Be careful when selecting an alias, because it is possible to select an alias that is only accessible by a subset of the servers that use the resource. If you select a global alias, you are guaranteed that an alias by that name is accessible to all users of the resource. If the alias has been overridden in a security domain, however, that alias is used instead of the global one. The tree view includes icons to help you select the proper alias:

- The alias is accessible by all servers that can access this resource.
- There is at least one server that cannot access the alias. Check the tree view to see if this is OK for the application that will be using this resource.
- The alias is defined in multiple places.

Considerations for isolated resource providers

There are some design considerations that you should be aware of when working with resource providers that you have specified to be isolated in their own class loaders.

Be aware of the following issues that you need to address if you isolate a resource provider in it's own class loader:

- **Client container**

  The client container does not manage the class path of resource providers, so resource providers that are isolated will not be supported in the client container.
- **Multiple resource provider versions per application**

  If an application refers to resources from multiple versions or implementations of the same resource provider, then all of the resource providers that are referenced must be isolated.
- **References to isolated resource provider classes**

  If a module directly refers to classes that are loaded by an isolated resource provider, which means the module has import statements of resource provider classes, the following restrictions are in place:
  - The module can only refer to resources from one version or implementation of an isolated resource provider. This is an inherent class loading restriction, because a module class loader can only refer to one version of a class.
  - The module cannot perform direct JNDI lookup without the use of Java EE resource reference meta-data. This restriction is required, because without resource reference metadata the application server has no mechanism to link the class loader of the module to the class loader of the isolated resource provider.

  The relational resource adapter does not generally allow direct access to resource provider classes, so these restrictions will typically only affect modules that implement the com.ibm.websphere.rsadapter.WSCallHelper class. For mail providers, these restrictions will most likely be in place, because the javax.mail API relies heavily on classes rather than interfaces. Therefore, the implementation details are necessarily part of the API.

Implicitly set client information

If you track client information in your database, you can choose one of two ways to pass WebSphere Application Server client data on database connections.

You can choose to *explicitly* pass the information about connections by calling an IBM proprietary API, setClientInformation(Properties), on the `com.ibm.websphere.rsadapter.WSConnection` object within your application code. The `com.ibm.websphere.rsadapter.WSConnection` object is located in the *plugins_root*`/com.ibm.ws.runtime.jar` file. In some cases, however, you might want WebSphere Application Server to handle the passing of client information to database connections. This method of setting the client information is referred to as *implicit*. You might choose the implicit method because:

- You want to keep your application free of proprietary APIs, or

- Your application uses container-managed persistence (CMP), in which case you cannot use the proprietary API to set client information on database connections.

The WebSphere Application Server trace facility provides the capability for setting client information implicitly. You can designate one of two special trace groups to enable or disable client information passing: “WAS.clientinfo trace” or “WAS.clientinfopluslogging trace” on page 343.

Possible runtime scenarios

- Connection sharing

  In the case of connection sharing, WebSphere Application Server sets the client information on the first acquired connection handle only. If connection sharing is enabled and two or more `getConnection` methods are called (resulting in two handles on the same connection), only the first `getConnection` call causes the client information to pass to the backend database. This scenario does not apply to the explicit process of passing client information; in such cases every `setClientInformation` method is relayed to the database regardless of connection sharing.
- Implicit/explicit co-existence

  When you use both the explicit and implicit procedures for relaying client information, some combination of the explicitly set data and implicitly set data is combined, but the explicit setting usually takes precedence. For example, if the application sets the client accounting information to "myAccountingInfo", the final accountingInfo string that is passed to the backend database looks something like the following sample code:

  ```
  000325_WSRdbManagedConnectionImpl@1234_myAccountingInfo:
  ```

  Where `000325` is the thread ID and `WSRdbManagedConnectionImpl@1234` is the WebSphere connection instance.
- Client information reset

  When you configure Application Server to pass client information, it does reset client information when a connection is returned to the pool, but *only* if the WAS.clientinfo and WAS.clientinfopluslogging trace mechanisms are disabled (that is, WAS.clientinfo=all=disabled:WAS.clientinfopluslogging=all=disabled).

  In the explicit case, however, the reset operation is done only when the application issues `setClientInformation(null)` on the WSConnection connection.

WAS.clientinfo trace

By default, the implicit mechanism is disabled. You can turn on this mechanism dynamically, without stopping and starting your application server, or statically by setting the WebSphere Application Server trace group *WAS.clientinfo=all=enabled*.

The information implicitly collected and set on the database connection consists of the *user name*, *user location* and *application name*.

Important: User name and user location can be implicitly collected and set only on the database connection if you enable Java 2 security.

user name
: The name of the user that initiates the application request. This option is collected and passed to the backend database (when supported). Information here is collected by calling the `WSSecurityHelper.getFirstCaller` method.

user location
: The name of the location of the user, in the form of cell:node:server. This option is collected and passed to the backend database (when appropriate). Information here is collected by calling the `WSSecurityHelper.getFirstServer` method.

application name
The name of the application running. This value is the output of the `getApplication` method from the Java EE Name object. This value is collected regardless of the Global Security setting.

WAS.clientinfopluslogging trace

When debugging database problems, such as deadlocks, there is a set of information that may help with the debugging effort. This information is typically obtained by enabling a WebSphere Relational Resource Adapter (RRA) trace, and an Enterprise JavaBeans (EJB) container trace. However, there are some cases where timing is an issue when reproducing a given problem. Having too much tracing information can alter the behavior of the application, such as change the timing, and the problem might no longer occur.

Because of this situation, a new trace group is provided where only a minimum set of information is collected. This trace group is WAS.clientinfopluslogging. This function sets the client information implicitly on the connection, just like the WAS.clientinfo trace, and, logs and traces important application activities. Those activities are:

- SQL strings that are run (such as, select userId from tabl1 where id=? for update).
- Start, commit, and rollback of transactions.
- EJB calls (such as, Create, Remove, findByPrimaryKey).

Enabling client information tracing with the administrative console

Use either of the methods outlined in this task to enable the passing and tracing of client information about a database connection.

About this task

Refer to the *Implicitly set client information* topic to determine which of the two available levels of client information passing and tracing is appropriate for your configuration. Once a level is selected, there are two methods of enabling it:

- Enable either of the WebSphere Application Server trace groups: WAS.clientinfo or WAS.clientinfopluslogging. Enabling either of these trace groups enables client information passing for all data sources of the application server. If client information is only required for a specific data source, consider using the method below.
- Create a data source custom property to enable client information tracing. This method is functionally equivalent to enabling WAS.clientinfo as described above, except that it is enabled only on the specified data source. A data source custom property does not exist to provide the equivalent functionality of the WAS.clientinfopluslogging trace group. If that level of tracing is required, use that trace group setting instead.

Procedure

1. Open the administrative console.
2. If you choose to enable client information passing and tracing by using a trace group:
   a. Select **Troubleshooting**.
   b. Select **Log and Trace**.
   c. Select the server you want to use.
   d. Select **Diagnostic Trace**.
   e. Select **Change log detail levels**.
   f. Select the **Configuration** or **Runtime** tab. Changes made to the Configuration are applied after a server restart. Changes made to the Runtime are applied immediately.
   g. In the **Trace Specification** entry field, type either `WAS.clientinfo=all` or `WAS.clientinfopluslogging=all`. To deactivate either trace, replace `=all` with `=off` (without spacing between characters) or delete the trace string entry.

3. If you choose to enable client information passing by using a data source custom property:
   a. Select **Resources**.
   b. Select **JDBC** > **Data sources**.
   c. Select the datasource on which you want to enable client information tracing.
   d. Select **Custom Properties** under the **Additional Properties** section.
   e. Press **New...**.
   f. Enter `enableClientInformation` in the **Name** field.
   g. Enter `true` in the **Value** field.
   h. Optional: Enter a description in the **Description** field, if desired.
   i. Select **java.lang.Boolean** in the **Type** field.
4. Press **OK**.
5. Save the changes to the configuration when prompted to do so.

About Apache Derby

The Apache Derby package that is bundled with the application server is backed by full IBM Quality Assurance (QA).

Note: WebSphere Application Server supports direct customer use of the Apache Derby database in *test* environments only. The product does not support direct customer use of Apache Derby database in *production* environments.

Unlike versions 5.1.60x and earlier, Apache Derby is a pure Java database server. The Apache Derby code base, which the open source community calls Derby, is a product of the Apache Software Foundation (ASF) open source relational database project. Apache Derby includes the Derby base code without any modification to the underlying source code. You can investigate more incompatibilities about Derby code at the Apache Derby website.

Note: Earlier versions of Apache Derby cannot conduct two phase-commit transactions over the Network Server framework, but later versions of the Derby Client JDBC driver provides Apache Derby with support for XA transactions. Only the Network Server framework provides support for multiple Java virtual machines (JVMs), such as application servers, to access Apache Derby.

Apache Derby is equipped with the following .bat/sh tools:
- sysinfo: displays database version information
- ij: manipulates the database instances

  transition: Use ij as an alternative for the Cloudscape cview tool, which does not exist in Derby. When you run the ij tool, surround the dbname by double quotation marks (" ") if it includes the full path name; for example:

  ```
  ij> connect '"c:\temp;create=true"'
  ```

  This is ' " " ' without spaces.
- dblook: dumps DDL information
- networkServerControl: controls the networkServer process (can be used for functions such as ping and trace)
- startNetworkServer: starts the networkServer process
- stopNetworkServer: stops the networkServer process

Attention: If you use non-English characters in your Derby database name, you need to update the ij script to specify the file encoding property to the JVM: -Dfile.encoding=*XXXXXX*, where *XXXXXX* is the encoding used to create the non-English characters.

Managing resources through JCA lifecycle management operations

You can manage the run-time status of your data source and connection factory resources to perform some data access administrative tasks without restarting the application server. This topic outlines the process for managing those resources through the administrative console.

Before you begin

When you manage the run-time status of connection factories or data sources, you are applying Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) lifecycle management operations to the MBeans that are associated with these resources. The management operations are PAUSE and RESUME. Pausing an MBean halts outbound communication to the backend, such as a database. This action affects all applications that use the corresponding connection factory or data source on the same server.

About this task

With these management operations, you can perform some administrative tasks dynamically, without restarting your application server:

- Respond to a security threat, and prevent new connection requests from reaching the backend
- Perform maintenance on the backend
- Apply configuration changes to non-required properties of the connection factory or data source, such as turning JDBC trace on or off, or modifying preferences for collecting client information

Procedure

1. Navigate to the administrative console page that corresponds to the resource type that you want to manage.
   - For connection factories, use either of the following paths:
     - **Resources** > **Resource Adapters** > **J2C connection factories**
     - **Resources** > **Resource Adapters** > **Resource Adapters** > *resource_adapter* > **J2C connection factories**
   - For data sources, use either of the following paths:
     - **Resources** > **JDBC** > **Data sources**
     - **Resources** > **JDBC** > **JDBC providers** > *JDBC_provider* > **Data sources**
2. Select the connection factory or data source configurations that you want to manage, and click **Manage state**. The administrative console now displays the JCA lifecycle management page, which contains a table that depicts the full scope configuration of your previous selection.

   The table is comprised of three columns:
   - **JNDI name:** The Java Naming and Directory Interface (JNDI) name of the connection factory or data source configuration.
   - **Running object scope:** The server that is running the connection factory or data source MBean.
   - **Status:** The status of the connection factory or data source MBean.
3. Select the rows that represent each invocation of the resource, per running server, that you want to manage. Be aware that when you click your management operation, WebSphere Application Server applies it to every resource object in your selection.

   Restriction: If the MBean status of a row has a value of NOT_ACCESSED, you cannot apply JCA lifecycle management operations to that MBean. The NOT_ACCESSED state indicates that the MBean exists on the specified server, but no applications performed a JNDI namespace lookup on the corresponding connection factory or data source.
4. Click **Pause** or **Resume**. The status column of the table changes to reflect the new state of the MBean.

JCA life cycle management

Use this page to perform JCA life cycle management operations on data source and connection factory MBeans. With these management operations, you can control the runtime status of the corresponding data source and connection factory resources.

You can view this administrative console page in different locations, depending on whether you want to manage data sources or J2C connection factories. For example:

- For connection factories: Click **Resources** > **Resource adapters** > **J2C connection factories**. Select the connection factory configurations that you want to manage, and click **Manage state**.
- For data sources, click **Resources** > **JDBC** > **Data sources**. Select the data source configurations that you want to manage, and click **Manage state**.

Guidelines for using this administrative console page:

- The table displays a list of MBeans that correspond to the data sources or connection factories in your selection from the previous console page. These MBeans are compatible with Version 6.0.2 and later of the application server.
- You can only perform JCA life cycle management actions on MBeans that are in the active state. In this context, an MBean is considered active when an application performs a Java Naming and Directory Interface (JNDI) name space lookup on the corresponding data source or connection factory resource.
- Pausing an MBean halts outbound communication to the backend, such as a database. This action affects all applications that use the resource on the selected server.

Pause:

Specifies to pause the MBean that is selected. Pausing an MBean halts outbound communication to the back-end resource, and this action affects all applications that use the resource on the selected server.

Note: When an application attempts a connection to a paused data source, the connection manager raises an SQLException with an error code of com.ibm.websphere.rsadapter.WSDataSource.ERROR_CONNECTION_POOL_IS_PAUSED.

Resume:

Specifies to resume the MBean that is selected. Resuming an MBean enables outbound communication to the back-end resource. This action affects all applications that use the resource on the selected server.

Purge:

Specifies to purge the contents of the connection pool for the data source or connection factory that is specified. Purging the pool does not affect ongoing transactions.

Name (JNDI name):

Specifies the name of the connection factory or data source configuration, followed by the Java Naming and Directory Interface (JNDI) name in parenthesis.

Running object scope:

Specifies the server that is running the connection factory or data source MBean.

Status:

Specifies the state of the connection factory or data source MBean.

Possible values:

State	Indications
ACTIVE	• The resource that corresponds with the MBean is ready to provide an application with connections to a backend. • An application performed a JNDI namespace lookup on this resource. You can apply the JCA life cycle management operation of PAUSE to an MBean in this state.
PAUSED	• All outbound communication to the backend through the corresponding resource is stopped, as a result of a JCA life cycle management operation that was applied previously to the MBean. • An application performed a JNDI namespace lookup on the resource. You can apply the JCA life cycle management operation of RESUME to an MBean in this state.
NOT_ACCESSED	• The MBean exists on the specified server, but no applications performed a JNDI name space lookup on the corresponding connection factory or data source. You cannot apply JCA life cycle management operations to an MBean in this state.

Chapter 8. Administering Dynamic caching

This page provides a starting point for finding information about the dynamic cache service, which improves performance by caching the output of servlets, commands, web services, and JavaServer Pages (JSP) files.

Dynamic caching features include replication of cache entries, cache disk offload, Edge-Side Include caching, web services, and external caching. Use external caching to control caches outside of the application server.

Administering the dynamic cache service

Using the dynamic cache service

Use the dynamic cache service to improve application performance by caching the output of servlets, web services, and WebSphere Application Server commands into memory.

Before you begin

Develop a cache policy for your application. The cache policy defines rules for what responses to cache and the amount of time the responses should be held in the cache. Refer to the Configuring cacheable objects with the cachespec.xml file article for more information.

About this task

The dynamic cache service is enabled by default. You can configure the default cache instance, as follows:

Procedure

1. Click **Servers > Server Types > WebSphere® application servers > *server_name* > Container services > Dynamic cache service**.
2. Configure the default cache instance or follow the links to enable servlet or portlet caching. Refer to the Dynamic cache service settings article for more information about default cache settings.

Example

This example puts all of the steps together for configuring the dynamic cache service with the `cachespec.xml` file, showing the use of the cache ID generation rules, dependency IDs, and invalidation rules.

Suppose that a servlet manages a simple news site. This servlet uses the query parameter "action" to determine if the request views (query parameter "view") news or updates (query parameter "update") news (used by the administrator). Another query parameter "category" selects the news category. Suppose that this site supports an optional customized layout that is stored in the user's session using the attribute name "layout". Here are example URL requests to this servlet:

- http://*yourhost/yourwebapp*/newscontroller?action=view&category=sports (Returns a news page for the sports category)
- http://*yourhost/yourwebapp*/newscontroller?action=view&category=money (Returns a news page for the money category)
- http://*yourhost/yourwebapp*/newscontroller?action=update&category=fashion (Allows the administrator to update news in the fashion category)

The following steps illustrate how to configure the dynamic cache service for this example with the `cachespec.xml` file:

1. Define the <cache-entry> elements that are necessary to identify the servlet. In this case, the URI for the servlet is "newscontroller", so this is the cache-entry <name> element. Because this example caches a servlet or JavaServer Pages (JSP) file, the cache entry class is "servlet".

```
<cache-entry>
<name> /newscontroller </name>
<class>servlet  </class>
 </cache-entry>
```

2. Define cache ID generation rules. This servlet caches only when action=view, so one component of the cache ID is the parameter "action" when the value equals "view". The news category is also an essential part of the cache ID. The optional session attribute for the user's layout is included in the cache ID. The cache entry is now:

```
<cache-entry>
 <name> /newscontroller </name>
 <class>servlet  </class>
  <cache-id>
  <component id="action" type="parameter">
   <value>view</value>
   <required>true</required>
  </component>
  <component id="category" type="parameter">
   <required>true</required>
  </component>
  <component id="layout" type="session">
   <required>false</required>
  </component>
 </cache-id>
</cache-entry>
```

3. Define dependency ID rules. For this servlet, a dependency ID is added for the category. Later, when the category is invalidated due to an update event, all views of that news category are invalidated. Following is an example of the cache entry after adding the dependency ID:

```
<cache-entry>
 <name>newscontroller </name>
 <class>servlet  </class>
  <cache-id>
  <component id="action" type="parameter">
   <value>view</value>
   <required>true</required>
  </component>
  <component id="category" type="parameter">
   <required>true</required>
  </component>
  <component id="layout" type="session">
   <required>false</required>
  </component>
 </cache-id>
 <dependency-id>category
  <component id="category" type="parameter">
   <required>true</required>
  </component>
 </dependency-id>
</cache-entry>
```

4. Define invalidation rules. Because a category dependency ID is already defined, define an invalidation rule to invalidate the category when action=update. To incorporate the conditional logic, add "ignore-value" components into the invalidation rule. These components do not add to the output of the invalidation ID, but only determine whether or not the invalidation ID creates and runs. The final cache-entry now looks like the following:

```
<cache-entry>
 <name>newscontroller </name>
 <class>servlet  </class>
  <cache-id>
  <component id="action" type="parameter">
   <value>view</value>
   <required>true</required>
```

```
 </component>
 <component id="category" type="parameter">
  <required>true</required>
 </component>
 <component id="layout" type="session">
  <required>false</required>
 </component>
</cache-id>
<dependency-id>category
 <component id="category" type="parameter">
  <required>true</required>
 </component>
</dependency-id>
<invalidation>category
 <component id="action" type="parameter" ignore-value="true">
  <value>update</value>
  <required>true</required>
 </component>
 <component id="category" type="parameter">
  <required>true</required>
    </component>
</invalidation>
</cache-entry>
```

What to do next

You might want to enable dynamic cache disk offload. This option moves cache entries that are expired from memory to disk for potential future access. Refer to the Configuring dynamic cache disk offload for more information about enabling disk offload.

Dynamic cache service settings

Use this page to configure and manage the dynamic cache service settings.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Container services > Dynamic cache service**.

Enable service at server startup:

The dynamic servlet cache service starts when the server starts.

Attention: This option displays on WebSphere Application Server V6.1 servers but is not available on WebSphere Application Server V7.0 servers.

Enable servlet caching:

The dynamic servlet cache service starts when servlet caching is enabled in Web Container panel.

Enable portlet caching:

Start the dynamic portlet cache service by enabling servlet caching, then, enabling portlet fragment caching under Portlet Container panel.

Cache provider:

Specifies whether to configure the server to use the default dynamic cache provider or an alternate cache provider. If an alternate cache provider is available, it appears in the list of available cache providers.

gotcha: If WebSphere eXtreme Scale is available as an alternate cache provider, see the topics "Introduction: Dynamic cache," and "Configuring dynamic cache (DynaCache) to use the

WebSphere eXtreme Scale dynamic cache provider" for more information about setting up and using WebSphere eXtreme Scale with WebSphere Application Server.

Cache size:

Specifies a positive integer as the value for the maximum number of entries that the cache holds.

Enter a cache size value in this field that is between the range of 100 through 200,000.

Default priority:

Specifies the default priority for cache entries, determining how long an entry stays in a full cache.

Default	1
Range	1 to 255

Limit memory cache size:

Specifies the size of the memory cache.

Use this feature to constrain the cache in terms of the JVM heap. In addition to specifying the cache size in MB, dynamic cache also enables you to set a high watermark and low watermark for the cache heap that is consumed. When the cache heap memory reaches the high watermark, dynamic cache either discards or is evicted from the disk using the least recently used (LRU) algorithm, until the cache is brought down to the low watermark. This functionality of limiting the cache in terms of the JVM heap is only available if the objects that are put into the cache implement the sizeable interface. This interface has one method that returns the size of the object in bytes put into the cache. Dynamic cache uses the sizeable interface to estimate the heap size of the cache.

Default	-1 to disable limiting the memory cache size
Range	1 to maximum integer

Memory cache size:

Specifies a value for the maximum memory cache size in megabytes (MB).

High threshold:

Specifies a high watermark when the memory cache eviction policy starts. The threshold is expressed in terms of the percentage of the memory cache size in megabytes (MB). The default value is 95%

Values	1 to 100

Low threshold:

Specifies a low watermark when the memory cache eviction policy ends. The threshold is expressed in terms of the percentage of the memory cache size in megabytes (MB). The default value is 80%.

Values	1 to 100

Enable disk offload:

Specifies whether disk offload is enabled.

By default, the dynamic cache maintains the number of entries that are configured in memory. If new entries are created while the cache is full, the priorities that are configured for each cache entry, and a least recently used algorithm, are used to remove entries from the cache. In addition to having a cache entry removed from memory when the cache is full, you can enable disk offload to have a cache entry copied to the file system (the location is configurable). Later, if that cache entry is needed, it is moved back to memory from the file system.

Before you enable disk offload, consider the following:

- You cannot specify the number of cache entries that are offloaded to disk.
- You cannot specify the amount of disk space to use.

Offload location:

Specifies the location on the disk to save cache entries when disk offload is enabled.

If disk offload location is not specified, the default location, `${WAS_TEMP_DIR}/node/server name/_dynacache/cache JNDI name` is used. If disk offload location is specified, the node, server name, and cache instance name are appended. For example, `${USER_INSTALL_ROOT}/diskoffload` generates the location as `${USER_INSTALL_ROOT}/diskoffload/node/server name/cache JNDI name`. This value is ignored if disk offload is not enabled.

The default value of the ${WAS_TEMP_DIR} property is `${USER_INSTALL_ROOT}/temp`. If you change the value of the ${WAS_TEMP_DIR} property after starting WebSphere Application Server, but do not move the disk cache contents to the new location:

- The application server creates a disk cache file at the new disk offload location.
- If the Flush to disk setting is enabled, all of the disk cache content at the old location is lost when you restart the application server

When you are specifying a directory, consider the following:

- If you use the default directory and the disk fills up, WebSphere Application Server could possibly stall if it must write messages to log files, and there is no more space.
- Depending on the operating system, you might see disk full messages on the console.

Flush to disk:

Specifies if in-memory cached objects are saved to disk when the server is stopped. This value is ignored if **Enable disk offload** is not selected.

Default	false

Limit disk cache size in GB:

Specifies a value for the maximum disk cache size in GB. When you select this option, you can specify a positive integer value. Leaving this option blank indicates an unlimited size. This setting applies only if enable disk offload is specified for the cache.

Value	3 and above.

Limit disk cache size in entries:

Specifies a value for the maximum disk cache size in number of entries. When you select this option, you can specify a positive integer value. Leaving this option blank indicates an unlimited size. This setting applies only if enable disk offload is specified for the cache.

Value	0 to MAXINT. A value of 0 indicates unlimited size.

Limit disk cache entry size:

Specifies a value for the maximum size of an individual cache entry in MB. Any cache entry larger than this value, when evicted from memory, is not offloaded to disk. When you select this option, you can specify a positive integer value. Leaving this option blank indicates an unlimited size. This setting applies only if enable disk offload is specified for the cache.

Value	0 to MAXINT. A value of 0 indicates unlimited size.

Disk cache performance settings:

Specifies the level of performance that is required by the disk cache. This setting applies only if **enableDiskOffload** is specified for the cache. Performance levels determine how memory resources should be used on background activity such as cache cleanup, expiration, garbage collection, and so on. This setting applies only if enable disk offload is specified for the cache.

High performance and high memory usage	Indicates that all metadata is kept in memory.
Balanced performance and balanced memory usage	Indicates some metadata is kept in memory. This is the default performance setting and provides an optimal balance of performance and memory usage for most users.
Low performance and low memory usage	Indicates that limited metadata is kept in memory.
Custom performance	Indicates that the administrator explicitly configures the memory settings that are used to support the above background activity. The administrator sets these values using the **DiskCacheCustomPerformanceSettings** object.

Disk cache cleanup frequency:

Specifies a value for the disk cache cleanup frequency, in minutes. If this value is set to 0, the cleanup runs only at midnight. This setting applies only when the Disk Offload Performance Level is low, balanced, or custom. The high performance level does not require disk cleanup, and this value is ignored.

Value	0 to 1440

Maximum buffer for cache identifiers per metaentry:

Specifies a value for the maximum number of cache identifiers that are stored for an individual dependency ID or template in the disk cache metadata in memory. If this limit is exceeded the information is offloaded to the disk. This setting applies only when the disk offload performance level is CUSTOM.

Value	100 to MAXINT

Maximum buffer for dependency identifiers:

Specifies a value for the maximum number of dependency identifier buckets in the disk cache metadata in memory. If this limit is exceeded the information is offloaded to the disk. This setting applies only when the disk cache performance level is custom.

Value	100 to MAXINT

Maximum buffer for templates:

Specifies a value for the maximum number of template buckets that are in the disk cache metadata in memory. If this limit is exceeded the information is offloaded to the disk. This setting applies only when the disk cache performance level is custom.

Value	10 to MAXINT

Disk cache eviction algorithm:

Specifies the eviction algorithm that the disk cache uses to evict entries when the high threshold is reached. This setting applies only if enable disk offload is specified for the cache. This setting does not apply when the disk cache eviction policy is set to none.

None	No eviction policy, so the disk cache can grow until it reaches its limit at which time the dynamic cache service stops writing to disk
Random	When the disk size reaches a high threshold limit, the disk cache garbage collector wakes up and randomly picks entries on the disk and evicts them until the size reaches a low threshold limit.
Size	When the disk size reaches a high threshold limit, the disk cache garbage collector wakes up and picks the largest entries on the disk and evicts them until the disk size reaches a low threshold limit.

High threshold:

Specifies when the eviction policy runs. The threshold is expressed in terms of the percentage of the disk cache size in GB or entries. The lower value is used when limit disk cache size in GB and limit disk cache size in entries are specified. This setting does not apply when the disk cache eviction policy is set to none.

Values	1 to 100

Low threshold:

Specifies when the eviction policy ends. The threshold is expressed in terms of the percentage of the disk cache size in GB or entries. The lower value is used limit disk cache size in GB and limit disk cache size in entries are specified. This setting does not apply when the disk cache eviction policy is set to none.

Values	1 to 100

Configuring dynamic cache (DynaCache) to use the WebSphere eXtreme Scale dynamic cache provider

The dynamic cache engine is the default cache provider for the dynamic cache APIs and framework. However, dynamic cache allows WebSphere eXtreme Scale to act as its core caching engine. You can configure dynamic cache to use WebSphere eXtreme Scale as your cache provider instead of the default dynamic cache engine. Configuring dynamic cache to use WebSphere eXtreme Scale lets you leverage transactional support, improved scalability, high availability, and other WebSphere eXtreme Scale features without changing your existing dynamic cache caching code.

Before you begin

- Read the "Introduction: Dynamic cache" topic for an overview of WebSphere eXtreme Scale functionality.
- Determine whether using WebSphere eXtreme Scale is beneficial to the applications running in your application servers.

  The WebSphere eXtreme Scale features significantly increase the distributed capabilities of the dynamic cache function beyond what is offered by the default dynamic cache engine and data replication service. With WebSphere eXtreme Scale, you can create caches that are truly distributed between multiple servers, rather than just replicate and synchronize caches between the servers. WebSphere eXtreme Scale caches are transactional and highly available, ensuring that each server sees the same dynamic cache content. WebSphere eXtreme Scale also offers a higher quality of service for cache replication than what is provided through the data replication service (DRS).

  However, these advantages do not mean that the eXtreme Scale dynamic cache provider is the right choice for every application. Use the decision trees and feature comparison matrix provided in the See the topic Configuring the dynamic cache provider for WebSphere eXtreme Scale in the WebSphere eXtreme Scale Version 7.0 Information Center for a overview of this cache provider.

If you decide that is beneficial for your applications, use the information provided in the "Configuring the dynamic cache provider for WebSphere eXtreme Scale" section of the *WebSphere eXtreme Scale Version 7.0 Product Overview* to determine the appropriate WebSphere eXtreme Scale topology for your caching deployment.

If you use WebSphere eXtreme Scale, instead of the default dynamic cache engine, dynamic cache has the following limitations:

- No disk cache support. The following custom properties do not work:
  - com.ibm.ws.cache.CacheConfig.enableDiskOffload
  - com.ibm.ws.cache.CacheConfig.diskOffloadLocation
  - com.ibm.ws.cache.CacheConfig.flushToDiskOnStop
  - com.ibm.ws.cache.CacheConfig.htodCleanupFrequency
  - com.ibm.ws.cache.CacheConfig.htodDelayOffload
  - com.ibm.ws.cache.CacheConfig.htodDelayOffloadEntriesLimit
  - com.ibm.ws.cache.CacheConfig.htodDelayOffloadDepIdBuckets
  - com.ibm.ws.cache.CacheConfig.htodDelayOffloadTemplateBuckets
  - com.ibm.ws.cache.CacheConfig.diskCachePerformanceLevel
  - com.ibm.ws.cache.CacheConfig.diskCacheEvictionPolicy
  - com.ibm.ws.cache.CacheConfig.diskCacheHighThreshold
  - com.ibm.ws.cache.CacheConfig.diskCacheLowThreshold
  - com.ibm.ws.cache.CacheConfig.diskCacheSize
  - com.ibm.ws.cache.CacheConfig.diskCacheSizeInGB
  - com.ibm.ws.cache.CacheConfig.diskCacheEntrySizeInMB
  - com.ibm.ws.cache.CacheConfig.explicitBufferLimitOnStop
  - com.ibm.ws.cache.CacheConfig.lruToDiskTriggerTime
  - com.ibm.ws.cache.CacheConfig.lruToDiskTriggerPercent
- No DRS replication (push or push-pull support). The following custom properties will not work:
  - com.ibm.ws.cache.CacheConfig.enableReplicationAcks
  - com.ibm.ws.cache.CacheConfig.enableCacheReplication
  - com.ibm.ws.cache.CacheConfig.replicationDomain
  - com.ibm.ws.cache.CacheConfig.cacheEntryWindow
  - com.ibm.ws.cache.CacheConfig.cachePercentageWindow

  - com.ibm.ws.cache.CacheConfig.cacheInvalidateEntryWindow
  - com.ibm.ws.cache.CacheConfig.cacheInvalidatePercentWindow
  - com.ibm.ws.cache.CacheConfig.filterTimeOutInvalidation
  - com.ibm.ws.cache.CacheConfig.filterLRUInvalidation
- The alias API feature is not supported for Object cache.
- Event listener is supported. When firing any event, WebSphere eXtreme Scale always sets the sourceOfInvalidation to `REMOTE`.
- Disable dependency ID, com.ibm.ws.cache.CacheConfig.disableDependencyId, and templates, com.ibm.ws.cache.CacheConfig.disableTemplatesSupport, are not supported.
- No PMI support.
- The following CacheStatistic counters are supported:
  - CacheHits
  - CacheLruRemoves
  - CacheMisses
  - CacheRemoves
  - ExplictInvalidationsFromMemory
  - MemoryCacheEntries
  - TimeoutInvalidationsFromMemory
- NioMap - skipMemoryAndWriteToDisk will not work because the disk cache is not supported. In addition, the DistributedNioMapObject.release() is not called to release byteBuffer to the NIO buffer management.

Complete the following actions to enable the WebSphere eXtreme Scale dynamic cache provider.

Procedure

1. Install the WebSphere eXtreme Scale client, or the WebSphere eXtreme Scale client and server packages in your application server for the remote server and the other topologies respectively.
2. Designate the WebSphere eXtreme Scale dynamic cache provider as your cache provider.

   Each cache instance can be individually configured to use WebSphere eXtreme Scale. The dynamic cache engine is the default cache provider for a cache instance. Cache instances configured with WebSphere eXtreme Scale can coexist with cache instances configured with DRS.

   Complete one of the following actions to designate the WebSphere eXtreme Scale dynamic cache provider as your cache provider:

   a. Use the administrative console to designate the WebSphere eXtreme Scale dynamic cache provider as your cache provider.
      1) In the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name*.
      2) Under Container Services, click **Dynamic cache service** *server_name*, and then, in the **Cache provider** field, select `WebSphere eXtreme Scale dynamic cache provider .`
      3) Click **OK**.

   b. Add the cacheProviderName property to the cacheinstances.properties file that is bundled in an enterprise application, and set the property to `com.ibm.ws.objectgrid.dynacache.CacheProviderImpl`.

      For example, for cache.instance.26, you would add the following line to the cacheinstances.properties file:

      ```
      cache.instance.26.cacheProviderName = com.ibm.ws.objectgrid.dynacache.CacheProviderImpl
      ```

   c. Add the following Factory APIs to the code for an enterprise application:

```
Properties p = new Properties();
               ==>p.put(CacheConfig.CACHE_PROVIDER_NAME, CacheConfig.CACHE_PROVIDER_OBJECT_GRID);
               DistributedMap map1 = DistributedMapFactory.getMap("myMap", p);
               DistributedMap map2 = DistributedObjectCacheFactory.getMap("myMap2", p);
```

3. Configure the replication setting for the cache instance.

   With the WebSphere eXtreme Scale dynamic cache provider you can have local cache instances, and replicated cache instances.

   If you are only going to use local cache instances, go to the last step, and save your configuration changes. In a local cache the WebSphere eXtreme Scale container is co-located with the JVM; that is, the WebSphere eXtreme Scale container and WebSphere Application Server share the same JVM heap.

   If you are going to use replicated caches, complete one of the following actions, depending on how you have created the cache instance:

   a. On the **Java virtual machine > Custom properties** page in the administrative console, click **New** again. Enter `com.ibm.ws.cache.CacheConfig.enableCacheReplication` in the **Name** field, and `true` in the **Value** field, and then click **OK**.

   b. Add the enableCacheReplication property to the cacheinstances.properties file that is bundled in an enterprise application, and set the property to `true`. .

      For example, for cache.instance.26, you would add the following line to the cacheinstances.properties file:

      ```
      cache.instance.26.enableCacheReplication = true
      ```

   c. Add the following Factory APIs to the code for an enterprise application:

      ```
      Properties p = new Properties();
                     p.put(CacheConfig.CACHE_PROVIDER_NAME, CacheConfig.CACHE_PROVIDER_OBJECT_GRID);
                     ==>p.put(CacheConfig.ENABLE_CACHE_REPLICATION, "true");
                     DistributedMap map1 = DistributedMapFactory.getMap("myMap", p);
                     DistributedMap map2 = DistributedObjectCacheFactory.getMap("myMap2", p);
      ```

4. Configure the WebSphere eXtreme Scale replication topology.

   The only required configuration parameter for the WebSphere eXtreme Scale dynamic cache provider is the cache topology parameter.

   a. On the **Java virtual machine > Custom properties** page in the administrative console, click **New** again. Enter `com.ibm.websphere.xs.dynacache.topology` in the **Name** field, and one of the following values in the **Value** field:

      ```
      embedded
      embedded_partitioned
      remote
      ```

      gotcha: If you specify `embedded_partitioned`, you must also add the com.ibm.websphere.xs.dynacache.num_initial_containers custom property to your JVM settings, and set this property to an integer that is equal to or slightly less than the total number of server instances that are accessing this distributed cache instance.

   b. Add the com.ibm.websphere.xs.dynacache.topology property to the cacheinstances.properties file that is bundled in an enterprise application, and set the property to `true`. .

      For example, for cache.instance.26, you would add the following line to the cacheinstances.properties file:

      ```
      cache.instance.26.enableCacheReplication = embedded
      ```

      gotcha: If you specify `embedded_partitioned`, you must also add the com.ibm.websphere.xs.dynacache.num_initial_containers property to the cacheinstances.propertiesfile, and set this property to an integer that is equal to or slightly less than the total number of server instances that are accessing this distributed cache instance.

For example, if a dynamic cache service is shared between grid members, then the variable should be set to the number of grid members.

c. Add the following Factory APIs to the code for an enterprise application:

```
Properties p = new Properties();
                p.put(CacheConfig.CACHE_PROVIDER_NAME, CacheConfig.CACHE_PROVIDER_OBJECT_GRID);
                p.put(CacheConfig.ENABLE_CACHE_REPLICATION, "true");
                ==>p.put("com.ibm.websphere.xs.dynacache.topology", "embedded");
                ==>p.put("com.ibm.websphere.xs.dynacache.num_initial_containers", "3");
                DistributedMap map1 = DistributedMapFactory.getMap("myMap", p);
                DistributedMap map2 = DistributedObjectCacheFactory.getMap("myMap2", p);
```

See the topic Configuring the dynamic cache provider for WebSphere eXtreme Scale in the WebSphere eXtreme Scale Version 7 Information Center for more information about the `embedded`, `embedded_partitioned`, and `remote` settings.

5. Configure the eXtreme Scale catalog service grid.

   When you run a catalog service grid, you must set the catalog.services.cluster custom property for the catalog service endpoints.

   See the topic Starting the catalog service process in a WebSphere Application Server environment in the WebSphere eXtreme Scale Version 7 Information Center for a description of how to start the catalog service process in a WebSphere Application Server environment.

   a. In the administrative console, click **System administration > Cell > Custom properties > New**.

   b. Enter `catalog.services.cluster` in the **Name** field, and the appropriate *server_name*:*host_name*:*client_port*:*peer_port*:*listener_port* value in the **Value** field.

      - *server_name* is the fully qualified name of the WebSphere process, such as the cell name, node name, or server name, of the server that hosts the catalog service. Example: `cellA\node1\nodeagent`
      - *host_name* is the name of the hosting server.
      - *client_port* is the port that is used for peer catalog grid communication.
      - *peer_port* is the port that is used for peer catalog grid communication.
      - *listener_port* is the listener port. This port must match the BOOTSTRAP_ADDRESS value that is defined in the server configuration.

      Following is an example of a valid value:

      ```
      cellA\node1\nodeagent:host.local.domain:6600:6601:2809,cellA\node2\
      nodeagent:host.foreign.domain:6600:6601:2809
      ```

   c. Click **OK**.

6. Click **Save** to save all of your configuration changes.
7. Restart all the application servers that you configured to use WebSphere eXtreme Scale.
8. Configure custom key objects.

   When you are using custom objects as keys, the objects must implement the Serializable or Externalizable interface. If you are using custom objects with the embedded or embedded partitioned topologies, you must place the objects on the shared library path, in the same way that you do if you are using the default dynamic cache provider. For more information, see the topic Using the DistributedMap and DistributedObjectCache interfaces for the dynamic cache.

   If you are using the remote topology, you must place the custom key objects on the CLASSPATH for the stand-alone WebSphere eXtreme Scale containers. See the steps to start a container process in the *WebSphere eXtreme Scale Version 7.0 Administration Guide* for more information. This publication is available on the WebSphere eXtreme Scale library page

9. Configure eXtreme Scale container servers.

   The cached data is stored in WebSphere eXtreme Scale containers. These containers can run inside or outside of a WebSphere Application Server process. The eXtreme Scale provider automatically creates containers inside a WebSphere Application Server process when you are using embedded or embedded partitioned topologies for a cache instance. No further configuration is needed for these topologies.

When you use the remote topology, you must start up stand-alone eXtreme Scale containers before you start the WebSphere Application Server instances that access the cache instance. The *WebSphere eXtreme Scale Version 7.0 Administration Guide* documents the steps that you need to complete to start stand-alone containers.

gotcha: Make sure that all the containers for a specific dynamic cache point to the same catalog service endpoints.

The dynacache-remoteobjectgrid. xml and dynacache-remote-definition.xml configuration files for the stand-alone eXtreme Scale Dynamic Cache provider containers are located in the *install_root*/customLibraries/ ObjectGrid/dynacache/etc directory if WebSphere eXtreme Scale is installed on top of the WebSphere Application Server, or in the *install_root*/ObjectGrid/dynacache/etc directory if you are using a stand-alone version of WebSphere eXtreme Scale. Make copies of these files to edit and use when launching stand-alone containers for the eXtreme Scale dynamic cache provider. The value specified for the numIntitialContainers parameter in the dynacache-remote-deployment.xml file should be based on the number of container processes being run.

The following example illustrates a UNIX-based command line entry that launches a stand-alone container for the WebSphere eXtreme Scale dynamic cache provider:

```
startOgServer.sh container1 -objectGridFile ../dynacache/etc/dynacache-remoteobjectgrid.
xml -deploymentPolicyFile ../dynacache/etc/dynacache-remotedeployment.
xml -catalogServiceEndpoints MyServer1.company.com:2809
```

gotcha: The set of container processes needs to have enough free memory to service all the dynamic cache instances configured to use the remote topology. Any WebSphere Application Server process that shares the same or equivalent values for catalog.services.cluster must use the same set of stand-alone containers. The number of containers and number of machines they reside on needs to be sized appropriately. See the topic Capacity planning and high availability in the *WebSphere eXtreme Scale Version 7.0 Product Overview* for additional details. This publication is available on the WebSphere eXtreme Scale library page.

10. Verify that the WebSphere eXtreme Scale dynamic cache provider is correctly configured.

    If the WebSphere eXtreme Scale dynamic cache provider is correctly configured, the system log contains a number of messages similar to the following messages:

    ```
    DYNA1001I: WebSphere Dynamic cache instance named "{0}" initialized successfully using cache provider "{1}".
    DYNA1071I: The cache provider \"{0}\" is being used.
    ```

    If the configuration and initialization of the cache instance with WebSphere eXtreme Scale fails the dynamic cache runtime reverts to the default dynamic cache cache provider, and you should see messages similar to the following message in the system log.

    ```
    DYNA1066E: Unable to initialize the cache provider \"{0}\".
    The Dynamic cache will be used to create the cache instance \"{1}\"
    instead of the configured cache provider.
    ```

Dynamic caching with Asynchronous Request Dispatcher:

Asynchronous Request Dispatcher (ARD) improves servlet response time when slow operations are logically separated and performed concurrently with other operations that are required to complete the response.

Servlet caching works with ARD include requests, with certain caveats. The dynamic caching service does not support the ESI and ExternalCache features when ARD is enabled, due to complex and intractable buffering issues with third party content.

If the include request has been cached by the dynamic cache service, the include request returns immediately with the response data written to the top-level response data for the servlet.

In combination with the Remote Request Dispatcher (RRD), include request processing can also be offloaded to other members in the application server core group, thus reducing resource requirements on the original application server.

Configuring servlet caching

After a servlet is invoked and completes generating the output to cache, a cache entry is created containing the output and the side effects of the servlet. These side effects can include calls to other servlets or JavaServer Pages (JSP) files or metadata about the entry, including timeout and entry priority information. Configure servlet caching to save the output of servlets and JavaServer Pages (JSP) files to the dynamic cache.

Before you begin

To enable servlet caching, you must complete the tasks in the Using the dynamic cache service topic.

About this task

Unique entries are distinguished by an ID string that is generated from the HttpServletRequest object each time the servlet runs. You can then base servlet caching on:

- Request parameters and attributes of the Universal Resource Identifier (URI) that was used to invoke the servlet
- Session information
- Other options, including cookies

Because JavaServer Pages files are compiled into servlets, the dynamic cache function treats JavaServer Pages files the same as servlets, except in specifically documented situations.

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > Web container settings > Web container** in the console navigation tree.
2. Select **Enable servlet caching** under the Configuration tab.
3. Click **Apply** or **OK**.
4. Restart WebSphere Application Server. Refer to the Managing application servers topic for more information.

What to do next

Define the cache policy for your servlets by Configuring cacheable objects with the cachespec.xml file.

Dynamic caching with Asynchronous Request Dispatcher:

Asynchronous Request Dispatcher (ARD) improves servlet response time when slow operations are logically separated and performed concurrently with other operations that are required to complete the response.

Servlet caching works with ARD include requests, with certain caveats. The dynamic caching service does not support the ESI and ExternalCache features when ARD is enabled, due to complex and intractable buffering issues with third party content.

If the include request has been cached by the dynamic cache service, the include request returns immediately with the response data written to the top-level response data for the servlet.

In combination with the Remote Request Dispatcher (RRD), include request processing can also be offloaded to other members in the application server core group, thus reducing resource requirements on the original application server.

Configuring portlet fragment caching

After a portlet is invoked and completes generating the output to cache, a cache entry is created, containing the output and the side effects of the portlet. These side effects can include calls to other portlets or metadata about the entry, including timeout and entry priority information. Configure portlet fragment caching with the WebSphere Application Sever administrative console to save the output of portlets to the dynamic cache.

Before you begin

To enable portlet fragment caching, you must complete the steps in the Using the dynamic cache service topic.

About this task

Unique entries are distinguished by an ID string that generates from the PortletRequest object each time the portlet runs. You can then base portlet fragment caching on:

- Request parameters and attributes
- Session information
- Portlet-specific information, portlet session, portlet window ID, portlet mode, and portlet window state

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > Portlet container settings > Portlet container** > **Application servers >** *server_name* in the administrative console navigation tree.
2. Select **Enable portlet fragment cache** under the Configuration tab.
3. Click **Apply** or **OK**.
4. Restart WebSphere Application Server.

   See the *Setting up the application serving environment* PDF for more information.

What to do next

Define a cache policy for your portlets. Note that portlets are not cached unless an applicable caching policy is defined in a cachespec.xml file. Refer to the Configuring cacheable objects with the cachespec.xml file topic for general task information about defining a cache policy. Refer to the Configuring caching policies for portlets topic for information about defining portlet-specific aspects in a cache policy.

Configuring caching policies for portlets:

Fragment caching for portlets requires that you define a cache policy in a cachespec.xml file, either within the portlet web application archives (WAR) file or globally. If no caching policy is defined and applicable to a particular portlet, that porlet is not cached.

WebSphere Application Server caching policies provide a lot of flexibility for defining cache IDs and invalidation rules that match the specific requirements of individual portlets. The caching policies that you can define are not necessarily compliant with the caching behavior that is defined by the Java Portlet Specification. The following sections provide some recommendations on how you can exploit the features of cachespec.xml file, to define a caching policy that conforms to the specification.

Cache expiration. Portlets define cache expiration time in the `<expiration-cache>` element of the portlet.xml deployment descriptor. If this element is not present, or has a value of zero, the portlet is not cached. The cache expiration time for portlets is only defined in the deployment descriptor; any cache timeout values that are specified in a cachespec.xml file have no effect.

Caching scopes. Portlets are defined in the <caching-scope> element of the portlet.xml deployment descriptor, whether the portlet content should be shared across all users or whether it contains user-specific information and must be cached individually for each user. To maintain this setting in your caching policy definition, include the com.ibm.wsspi.portletcontainer.user_cache_scope attribute in your cache key, with the following cache key component:

```
<component id="com.ibm.wsspi.portletcontainer.user_cache_scope" type="attribute"/>
```

This attribute has the following values:

- The value, public, in a portlet that defines public cache scope.
- The current logon user ID in a portlet that defines private cache scope.
- Null (anonymous) in a portlet that defines private cache scope if no user is logged on.

If you want to cache portlet content for anonymous access, even in a portlet that defines private cache scope, add <required>false</required> to the cache key component. This implies that all anonymous browser access will retrieve the same cache content.

Portlet lifecycle methods. The Java Portlet Specification defines the four lifecycle phases: action, event, render and resource for running in a portlet. Only the render and resource phases produce content; the action and event phases invoke portlet activity without generating content and must not be cached. The lifecycle phase for a portlet call is available in the javax.portlet.lifecycle_phase request attribute. Check for the correct lifecycle by including the following cache key component:

```
<component id="javax.portlet.lifecycle_phase" type="attribute">
 <value>RENDER_PHASE</value>
</component>
```

This cache key component only caches render requests to the portlet. Cache additional resource requests by adding the RESOURCE_PHASE. In many cases, the best approach is to define a separate <cache-id> element for resource requests. The resource ID is available in the com.ibm.wsspi.portletcontainer.resource_id request attribute for caching key generation in resource requests.

Request parameters. Portlets can typically display multiple views. Render parameters distinguish which view displays. Each combination of parameters addresses a different view of the portlet. All views need to be cached separately; therefore, the full request parameter map should normally be included in the cache key. The com.ibm.wsspi.portletcontainer.all_parameters attribute provides a unique value for the content of the full request parameter map that can be used with the following cache key component:

```
<component id="com.ibm.wsspi.portletcontainer.all_parameters" type="attribute">
 <required>false</required>
</component>
```

If you write a cache policy for a specific portlet, and you know exactly which views of the portlet are addressed by which request parameters, it is usually more efficient to use specific <parameter> elements in the cache key to cache only the most important views of the portlet.

Other cache key components. Depending on your usage scenario, you will need to include other information in your cache key, if the returned content depends on it (for example, the portlet mode and window state, or the request locale in a multi-language portal). In a multi-device portal that supports different markup types, the returned content type should also be part of the cache key. The content type for a portlet is available in the com.ibm.wsspi.portletcontainer.response_contenttype request attribute.

Cache invalidation. The Java Portlet Specification states that action and event requests to a portlet must invalidate all currently cached content. The portlet caching definitions usually allow for caching multiple views of a portlet at the same time. To invalidate all of them, use the dependency ID mechanism of cachespec.xml.

Define a common dependency ID for all views that should be invalidated by an action. The common ID will usually only include the portlet window ID and the user scope, so that a portlet action does not affect private cache entries for other users:

```
<dependency-id>action
 <component id="" type="portletWindowId"/>
 <component id="com.ibm.wsspi.portletcontainer.user_cache_scope" type="attribute"/>
</dependency-id>
```

Define an invalidation rule that repeats the dependency ID and adds the current lifecycle method as a condition. It is essential to have the `ignore-value` attribute on the condition part. The lifecycle attribute must not be part of the returned invalidation ID, because that invalidation ID must match exactly with the dependency ID that is specified above.

```
<invalidation>action
 <component id="" type="portletWindowId"/>
 <component id="com.ibm.wsspi.portletcontainer.user_cache_scope" type="attribute"/>
 <component id="javax.portlet.lifecycle_phase" type="attribute" ignore-value="true">
  <value>ACTION_PHASE</value>
  <value>EVENT_PHASE</value>
 </component>
</invalidation>
```

Following the same pattern, specify more complex invalidation rules in caching policies for individual portlets, (for example, you can only invalidate a subset of the portlet views for specific actions that are determined by request parameters). The following example code describes a generic caching configuration that conforms to the behavior that is defined by the Java Portlet Specification:

Sample cachespec.xml file

```
<?xml version="1.0" ?>
<!DOCTYPE cache SYSTEM "cachespec.dtd">
<cache>
 <cache-entry>
  <class>portlet</class>
  <name>MyPortlet</name>
  <property name="consume-subfragments">true</property>
  <cache-id>
   <component id="" type="portletWindowId"/>
   <component id="com.ibm.wsspi.portletcontainer.user_cache_scope" type="attribute"/>

   <component id="" type="portletWindowState">
   <!-- minimized portlets are not cached -->
   <not-value>minimized</not-value>
  </component>
  <component id="" type="portletMode"/>

  <component id="" type="locale"/>
  <component id="com.ibm.wsspi.portletcontainer.response_contenttype" type="attribute"/>

  <component id="com.ibm.wsspi.portletcontainer.all_parameters" type="attribute">
   <required>false</required>
  </component>

  <component id="javax.portlet.lifecycle_phase" type="attribute">
   <value>RENDER_PHASE</value>
  </component>
 </cache-id>

 <dependency-id>action
  <component id="" type="portletWindowId"/>
  <component id="com.ibm.wsspi.portletcontainer.user_cache_scope" type="attribute"/>
 </dependency-id>

 <invalidation>action
  <component id="" type="portletWindowId"/>
```

```
  <component id="com.ibm.wsspi.portletcontainer.user_cache_scope" type="attribute"/>
  <component id="javax.portlet.lifecycle_phase" type="attribute" ignore-value="true">
   <value>ACTION_PHASE</value>
   <value>EVENT_PHASE </value>
  </component>
 </invalidation>

 </cache-entry>
</cache>
```

Configuring portlet fragment caching with the wsadmin tool

You can configure portlet fragment caching with scripting and the wsadmin tool.

Before you begin

Before starting this task, the wsadmin tool must be running. See the *Using the administrative clients* PDF for more information.

About this task

Important: If you use the wsadmin tool to enable portlet fragment caching, you must make sure that servlet caching is also enabled. Similarly if you use the wsadmin tool to disable portlet fragment caching, you must make sure that servlet caching is also disabled. The settings for these two caching functions must stay synchronized. If you enable or disable portlet fragment caching using the administrative console, synchronization is automatically taken care of for you.

Procedure

1. Locate the server object. The following example selects the first server found:

   Using Jacl:

   ```
   set s1 [$AdminConfig getid /Server:server1/]
   ```

   Using Jython:

   ```
   s1 = AdminConfig.getid('/Server:server1/')
   ```

2. List the web containers and assign them to the `wc` variable, for example:

   Using Jacl:

   ```
   set wc [$AdminConfig list PortletContainer $s1]
   ```

   Using Jython:

   ```
   wc = AdminConfig.list('PortletContainer', s1)
   ```

3. Set the enablePortletCaching attribute to `true` and assign it to the `serEnable` variable, for example:

   Using Jacl:

   ```
   set serEnable "{enablePortletCaching true}"
   ```

   Using Jython:

   ```
   serEnable = [['enablePortletCaching', 'true']]
   ```

4. Enable caching, for example:

   Using Jacl:

   ```
   $AdminConfig modify $wc $serEnable
   ```

   Using Jython:

   ```
   AdminConfig.modify(wc, serEnable)
   ```

Configuring caching for Struts and Tiles applications

Use this task to cache Struts and Tiles applications.

Before you begin

Before you configure Struts and Tiles caching, you should have a developed application. Find more information about developing Struts and Tiles applications on the Apache Struts Web Application Framework on the Apache website at http://struts.apache.org/.

About this task

Use this task when you want to cache data in Struts and Tiles applications.

Struts is an open source framework for building web applications using the Model-View-Controller (MVC) architecture. The Struts framework has a controller component and integrates with other technologies to provide the model and the view. Struts provide a control layer for the web application, which reduces construction time and maintenance costs.

The Tiles framework builds on the jsp:include feature and is bundled with the Struts web application framework. The Tiles framework reduces the duplication between JavaServer Pages (JSP) files and makes website layouts flexible and easy to maintain by assembling presentation pages from component parts.

Struts and Tiles caching is an extension of servlet and JSP caching, so the actions performed for each type of caching are very similar. Refer to the Configuring servlet caching topic for more information about servlet caching.

Procedure

1. Enable servlet and JSP caching. Enabling servlet caching automatically enables Struts and Tiles caching. Refer to the Configuring servlet caching topic for more information about servlet caching.
2. Develop the cache policy. A cache policy is required to cache a struts or tiles response.

   To develop a Struts cache policy:

   The Struts framework provides the controller component in the MVC-style application. The controller is a servlet called `org.apache.struts.action.ActionServlet.class`. In the `web.xml` file of the application, a servlet mapping of `*.do` is added for this Struts ActionServlet servlet so that every request for a Web address that ends with `.do` is processed. The ActionServlet servlet uses the information in the `struts-config.xml` file to decide which Struts action class runs the request for the specified resource.

   In the previous version of WebSphere Application Server, only one cache policy per servlet was supported. However, when you are using Struts, every request that ends in `.do` maps to the same ActionServlet servlet. To cache Struts responses, write a cache policy for the ActionServlet servlet based on its servlet path.

   For example, consider two Struts actions: `/HelloParam.do` and `/HelloAttr.do`. To cache the responses based on the id request parameter and the arg request attribute respectively, use the following cache policy:

```
<cache-entry>
 <class>servlet</class>
 <name>org.apache.struts.action.ActionServlet.class</name>
 <cache-id>
  <component id="" type="servletpath">
   <value>/HelloParm.do</value>
  </component>
 </cache-id>
 <cache-id>
  <component id="" type="servletpath">
   <value>/HelloAttr.do</value>
  </component>
  <component id="arg" type="attribute">
```

```
   <required>true</required>
  </component>
 </cache-id>
</cache-entry>
```

With the current version of WebSphere Application Server, you can map multiple cache policies for a single servlet. You can rewrite the previous cache policy as in the following example:

```
<cache-entry>
 <class>servlet>
 <name>/HelloParam.do</name>
 <cache-id>
  <component id="id" type="parameter">
   <required>true</required>
  </component>
</cache-entry>
<cache-entry>
 <class>servlet</class>
 <name>/HelloAttr.do</name>
 <cache-id>
  <component id="arg" type="attribute">
   <required>true</required>
  </component>
 </cache-id>
</cache-entry>
```

To develop a Tiles cache policy:

The Tiles framework is built on the jsp:include tag, so everything that applies to JSP caching also applies to Tiles. You must set the flush attribute to `true` in any fragments that are included using the tiles:insert tag for the fragments to be cached correctly. The extra feature in tiles caching over JSP caching is based on the tiles attribute. For example, you might develop the following layout.jsp template:

```
<html>
 <%String categoryId = request.getParameter("categoryId")+"test"; %>
 <tiles:insert attribute="header">
  <tiles:put name="categoryId" value="<%= categoryId %>" />
 </tile:insert>
 <table>
  <tr>
   <td width="70%" valign="top"><tiles:insert attribute="body" /> </td>
  </tr>
  <tr>
   <td colspan="2"><tiles:insert attribute="footer" /></td>
  </tr>
 </table>
</body>
</html>
```

The nested tiles:put tag specifies the attribute of the inserted tile. In the `layout.jsp` template, the categoryId attribute is defined and passed on to the tile that is inserted into the placeholder for the header. In the following example, the `layout.jsp` file is inserted into another JSP file:

```
<html>
<body>
<tiles:insert page="layout.jsp?categoryId=1002" flush="true">
 <tiles:put name="header" value="/header.jsp" />
 <tiles:put name="body" value="/body.jsp" />
 <tiles:put name="footer" value="/footer.jsp" />
</tiles:insert>
</body>
</html>
```

The categoryId tile attribute is passed on to the header.jsp file. The header.jsp file can use the <tiles:useAttribute> tag to retrieve the value of categoryId. To cache the header.jsp file based on the value of the categoryId attribute, you can use the following cache policy:

```
<cache-entry>
 <class>servlet</class>
 <name>/header.jsp</name>
 <cache-id>
  <component id="categoryId" type="tiles_attribute">
  <required>true</required>
  </component>
 </cache-id>
</cache-entry>
```

3. Ensure your cache policy is working correctly. You can modify the policies within the cachespec.xml file while your application is running. Refer to the Configuring cacheable objects with the cachespec.xml file topic for more information about cache policies.

Results

What to do next

Refer to the Task overview: Using the dynamic cache service to improve performance for more information about the dynamic cache.

Configuring dynamic cache disk offload

Use this task to configure dynamic cache disk offload, which saves cache entries that are deleted from the memory cache to disk.

About this task

By default, when the number of cache entries reaches the configured limit for a given application server, cache entries are removed from the memory cache, allowing newer entries to be stored in the cache. Use disk offload to copy the cache entries that are being removed from the memory cache to disk for potential future access.

Procedure

1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > Container services > Dynamic cache service** .
2. Select **Enable disk offload**.
3. After you enable the disk offload, you can set the Disk offload location. The disk offload location specifies where to save the cache entries on the disk. The disk offload location must be unique for any application servers that are defined on the same node. If you have multiple servers defined on the same node, make sure the disk offload location is different for each server.
4. Enable Flush to disk if you want cache objects that are in memory to be saved to disk when the server is stopped. Disk offload must be enabled if you choose this option. If you do not enable flush to disk, all the cache objects are deleted when the server stops.
5. Click **Apply** or **OK**.
6. Restart WebSphere Application Server.

Results

You enabled disk offload. Memory cache entries are moved to disk for potential future access.

When you have two or more application servers with servlet caching enabled and the application servers specify the same disk offload location for their caches through the dynamic cache service, the following exceptions might occur:

```
java.lang.NullPointerException
       at com.ibm.ws.cache.CacheOnDisk.readTemplate(CacheOnDisk.java:686)
       at com.ibm.ws.cache.Cache.internalInvalidateByTemplate(Cache.java:828)
```

or:

```
java.lang.NullPointerException
       at com.ibm.ws.cache.CacheOnDisk.readCacheEntry(CacheOnDisk.java:600)
       at com.ibm.ws.cache.Cache.getCacheEntry(Cache.java:341)
```

If one server is run as root and the other servers are run as non-root, this problem could occur. For example, if `server1` runs as root and `server2` runs as `wasuser` or `wasgroup`, the cache files in the disk offload location might be created with root permissions. This situation causes the applications running on the non-root servers to crash when they try to read or write to the cache.

Java virtual machine cache settings:

Use this page to set Java virtual machine (JVM) custom properties to maintain cache entries that are saved to disk.

You can set the custom properties globally to affect all cache instances, or you can set the custom property on a single cache instance. In most cases, set the properties on the individual cache instances. To set the custom properties on the default cache instance, use the global option. If you set the same property both globally and on a cache instance, the value that is set on the cache instance overrides the global value.

To configure the custom properties on a single object cache instance or servlet cache instance, perform the following steps:

1. In the administrative console, click one of the following paths:
   - To configure a servlet cache instance, click **Resources > Cache instances > Servlet cache instances >** *servlet_cache_instance_name* **> Custom properties > New**.
   - To configure an object cache instance, click **Resources > Cache instances > Object cache instances >** *object_cache_instance_name* **> Custom properties > New**.
2. Type the name of the custom property. When configuring these custom properties on a single cache instance, you do not use the full property path. For example, type `explictBufferLimitOnStop` to configure the `com.ibm.ws.cache.CacheConfig.explictBufferLimitOnStop` custom property.
3. Type a valid value for the property in the **Value** field.
4. Save the property and restart WebSphere Application Server.

To configure the custom property globally across all configured cache instances, perform the following steps:

1. In the administrative console, click **Servers > Application servers >** *server_name* **> Java and process management > Process definition > Java virtual machine > Custom properties > New**.
2. Type the name of the custom property (`com.ibm.ws.cache.CacheConfig.explictBufferLimitOnStop`) in the **Name** field.
3. Type a valid value for the property in the **Value** field.
4. Save the property and restart WebSphere Application Server.

Also use these properties to tune the delay offload function for the disk cache.

Important: Setting these custom properties using the **wsadmin** command is deprecated for WebSphere Application Server Version 7.0. Use the administrative console to set these properties. The individual property descriptions include information on how to use the administrative console to set these properties.

The delay offload function uses extra memory buffers for dependency IDs and templates to delay the disk offload and minimize the input and output operations. However, if most of your cache IDs are longer than 100 bytes, the delay offload function might use too much memory. Use any combination of the following properties to tune your configuration:

- To increase or decrease the in-memory limit of cache IDs for dependency ID and template buffers, use the `com.ibm.ws.cache.CacheConfig.htodDelayOffloadEntriesLimit` custom property.
- To disable the disk cache delay offload function, use the `com.ibm.ws.cache.CacheConfig.htodDelayOffload` custom property. Disabling this property saves all cache entries to disk immediately after removing them from the memory cache.

You can define the following Java virtual machine cache settings:

- "com.ibm.ws.cache.CacheConfig.explictBufferLimitOnStop"
- "com.ibm.ws.cache.CacheConfig.htodCleanupFrequency"
- "com.ibm.ws.cache.CacheConfig.htodDelayOffloadEntriesLimit" on page 371
- "com.ibm.ws.cache.CacheConfig.lruToDiskTriggerPercent" on page 371
- "com.ibm.ws.cache.CacheConfig.lruToDiskTriggerTime" on page 372

com.ibm.ws.cache.CacheConfig.explictBufferLimitOnStop:

Use this custom property when the flush-to-disk-on-stop feature is enabled. When the server is stopping, offloads are limited to the value specified for this property, pending removal of entries in the explicit invalidation buffer.

If this property is set to `0`, there is no limit to the number of offloads that can occur. Only positive integers are accepted as values for this property. If the number of entries in the explicit invalidation buffer is greater than the specified limit, all of the disk files for this specified cache instance are deleted after the server stops.

Important: You cannot use the administrative console to set this property.

com.ibm.ws.cache.CacheConfig.htodCleanupFrequency:

Use this property to change the amount of time between disk cache cleanup.

Important: Setting this custom property manually is deprecated for V6.1. Therefore, you should use the administrative console to set this property. To set this property in the administrative console, click one of the following paths:

- To configure a servlet cache instance, click **Resources > Cache instances > Servlet cache instances >** *servlet_cache_instance_name*.
- To configure an object cache instance, click **Resources > Cache instances > Object cache instances >** *object_cache_instance_name*.

Then:

1. Under Disk Cache setting, select the Enable disk offload field if it is not already selected.
2. Under Performance Settings, select Balanced performance and balanced memory usage or Custom.
3. In the Disk cache cleanup frequency field, specify an appropriate length of time, in minutes.

By default, the disk cache cleanup is scheduled to run at midnight to remove expired cache entries and cache entries that have not been accessed in the past 24 hours. However, if you have thousands of cache entries that might expire within one or two hours, the files that are in the disk cache can grow large and

become unmanageable. Use the `com.ibm.ws.cache.CacheConfig.htodCleanupFrequency` custom property to change the time interval between disk cache cleanup.

Units	minutes For example, a value of 60 means 60 minutes between each disk cache cleanup.
Default	0 The disk cache cleanup occurs at midnight every 24 hours.

com.ibm.ws.cache.CacheConfig.htodDelayOffloadEntriesLimit:

Use this property to specify the number of different cache IDs that can be saved in memory for the dependency ID and template buffers. Consider increasing this value if you have a lot of memory in your server and you want to increase the performance of your disk cache.

Important: Setting this custom property using the **wsadmin** command is deprecated for V7.0. Therefore, you should use the administrative console to set this property. To set this property in the administrative console, click one of the following paths:

- To configure a servlet cache instance, click **Resources > Cache instances > Servlet cache instances >** *servlet_cache_instance_name*.
- To configure an object cache instance, click **Resources > Cache instances > Object cache instances >** *object_cache_instance_name*.

Then:

1. Under Disk Cache setting, select the Enable disk offload field, if it is not already selected.
2. Under Disk Cache settings, select Limit disk cache size in entries, if it is not already selected.
3. In the Disk cache size field, specify the number of cache IDs that can be saved in memory for the dependency ID and template buffers.

Units	number of cache IDs For example, a value of 1000 means that each dependency ID or template ID can have up to 1000 different cache IDs in memory.
Default	1000
Minimum	100

com.ibm.ws.cache.CacheConfig.lruToDiskTriggerPercent:

Use this custom property to set the percentage of the memory cache size to be used as an overflow buffer when disk offload is enabled.

Cache entries in the overflow buffer are purged and asynchronously offloaded to disk at a frequency of *lruToDiskTriggerTime* milliseconds. If the memory overflow buffer is full, cache entries are offloaded to disk synchronously on the thread for the caller.

Units	integer, percentage
Lower bound	0
Upper bound	100

Scope	Configurable per cache instance.

com.ibm.ws.cache.CacheConfig.lruToDiskTriggerTime:

Use this custom property to set the frequency with which cache entries in memory are asynchronously offloaded to disk when the disk offload feature is enabled.

Units	integer, milliseconds
Lower bound	0
Upper bound	5000
Scope	Applicable to all cache instances.

Configuring Edge Side Include caching

The web server plug-in contains a built-in ESI processor. The ESI processor can cache whole pages, as well as fragments, providing a higher cache hit ratio. The cache implemented by the ESI processor is an in-memory cache, not a disk cache, therefore, the cache entries are not saved when the web server is restarted.

About this task

Edge Side Include (ESI) is configured through the `plugin-cfg.xml` file.

When a request is received by the web server plug-in, it is sent to the ESI processor, unless the ESI processor is disabled. It is enabled by default. If a cache miss occurs, a Surrogate-Capabilities header is added to the request and the request is forwarded to the WebSphere Application Server. If servlet caching is enabled in the application server, and the response is edge cacheable, the application server returns a Surrogate-Control header in response to the WebSphere Application Server plug-in.

The value of the Surrogate-Control response header contains the list of rules that are used by the ESI processor to generate the cache ID. The response is then stored in the ESI cache, using the cache ID as the key. For each ESI include tag in the body of the response, a new request is processed so that each nested include results in either a cache hit or another request that forwards to the application server. When all nested includes have been processed, the page is assembled and returned to the client.

The ESI processor is configurable through the WebSphere web server plug-in configuration file `plugin-cfg.xml`. The following is an example of the beginning of this file, which illustrates the ESI configuration options.

```
<?xml version-"1.0"?>
<Config>
  <Property Name="esiEnable" Value="true"/>
  <Property Name="esiMaxCacheSize" Value="1024"/>
  <Property Name="esiInvalidationMonitor" Value="false"/>
```

Procedure

- The first option, esiEnable, can be used to disable the ESI processor by setting the value to false. ESI is enabled by default. If ESI is disabled, then the other ESI options are ignored.
- The second option, esiMaxCacheSize, is the maximum size of the cache in 1K byte units. The default maximum size of the cache is 1 megabyte.

  If the first response has a Content-Length response header, the web server plug-in checks for the response size. If the size of the response body is larger than the available ESI caching space, the response passes through without being handled by ESI.

  Some parent responses have nested ESI includes. If a parent response is successfully stored in the ESI cache, and any subsequent nested include has a Content-length header that specifies a size larger than

the available space in the ESI cache, but smaller than the value specified for esiMaxCacheSize property, the plug-in ESI processor evicts other cache elements until there is enough space for the nested include in the ESI cache.

- The third option, esiInvalidationMonitor, specifies if the ESI processor should receive invalidations from the application server. ESI works well when the web servers following a threading model are used, and only one process is started. When multiple processes are started, each process caches the responses independently and the cache is not shared. This could lead to a situation where, the system's memory is fully used up by ESI processor. There are three methods by which entries are removed from the ESI cache: first, an entry expiration timeout occurs; second, an entry is purged to make room for newer entries; or third, the application server sends an explicit invalidation for a group of entries. For the third mechanism to be enabled, the esiInvalidationMonitor property must be set to true and the DynaCacheEsi application must be installed on the application server. The DynaCacheEsi application is located in the installableApps directory and is named DynaCacheEsi.ear. If the ESIInvalidationMonitor property is set to true but the DynaCacheEsi application is not installed, then errors occur in the web server plug-in and the request fails.
- This ESI processor is monitored through the CacheMonitor application. For the ESI processor cache to be visible in the CacheMonitor, the DynaCacheEsi application must be installed as described above, and the ESIInvalidationMonitor property must be set to true in the `plugin-cfg.xml` file.
- When WebSphere Application Server is used to serve static data, such as images and HTML on the application server, the URLs are also cached in the ESI processor. This data has a default timeout of 300 seconds. You can change the timeout value by adding the property com.ibm.servlet.file.esi.timeOut to the Java virtual machine (JVM) command line parameters. The following example shows how to set a one minute timeout on static data cached in the plug-in:

  ```
  -Dcom.ibm.servlet.file.esi.timeOut=60
  ```

  For information about configuring alternate URL, see the *Tuning guide* PDF.

Configuring alternate URL:

Alternate URL is a method for edge caching JavaServer Pages (JSP) files and servlet responses that you can not request externally. Dynamic cache provides support to recognize the presence of an Edge Side Include (ESI) processor and to generate ESI include tags and appropriate cache policies for edge fragments that can be cached. However, you must be able to externally request an edge fragment from the application server before it can be cached. In other words, if a user types the URL in their browser with the appropriate parameters and cookies for the fragment, WebSphere Application Server must be able to return the content for that fragment.

About this task

One of the standard Java Platform, Enterprise Edition (Java EE) programming architectures is the model-view-controller (MVC) architecture, where a call to a controller servlet might include one or more child JSP files to construct the view. When using the MVC programming model, the child JSP files are edge cached only if you can request these JSP files externally, which is not usually the case. For example, if a child JSP file uses one or more request attributes that are determined and set by the controller servlet, you cannot cache that JSP file on the edge. You can use alternate URL support to overcome this limitation by providing an alternate controller servlet URL used to invoke the JSP file.

The alternate URL for a JSP file or a servlet is set in the `cachespec.xml` file as a property with the name `alternate_url`. You can set the alternate URL either on a per cache-entry basis or on a per cache-id basis. It is valid only if the `EdgeCacheable` property is also set for that entry. If the `EdgeCacheable` property is not set, the `alternate_url` property is ignored. The following is a sample cache policy using the `alternate_url` property:

```
<cache-entry>
    <class>servlet</class>
    <name>/AltUrlTest2.jsp</name>
```

```
    <property  name="EdgeCacheable">true</property>
    <property  name="alternate_url">/alturlcontroller2</property>
        <cache-id>
            <timeout>600</timeout>
            <priority>2</priority>
        </cache-id>
</cache-entry>
```

What to do next

For more information on the `cachespec.xml` file, refer to the cachespec.xml file topic.

Configuring external cache groups

The dynamic cache can control caches outside of the application server, such as the Edge server, an IBM HTTP Server, or an HTTP Server ESI Fragment Processor plug-in.

About this task

When external cache groups are defined, the dynamic cache matches externally cacheable cache entries with those groups, and pushes cache entries and invalidations out to those groups. This allows WebSphere Application Server to manage dynamic content beyond the application server. The content can then be served from the external cache, instead of the application server, improving savings in performance.

Procedure

1. Open the administrative console.
2. Enable the dynamic cache.
   a. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > Container services > Dynamic cache service**.
   b. Select **Enable service at server startup** to enable the dynamic cache each time the application server starts.
3. Define the external cache group that WebSphere Application Server should control.
   a. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name* > Container services > Dynamic cache service > External cache groups**.
   b. Click **New** or choose an external cache group from the list.
4. Configure cache group members.
   a. Click **External cache groups** from the dynamic cache administrative console page. Then click **New** or choose an external cache group from the list.
   b. Click **External cache group members > New** or choose an external cache group member from the list.
   c. Type the configuration string in the Address field.
   d. Type the adapter bean name in the Adapter Bean Name field.
   e. Save the configuration.
   f. Click **Apply** or **OK**.

External cache group collection:

Use this page to define sets of external caches that are controlled by WebSphere Application Server on web servers such as IBM Edge Server and IBM HTTP Server.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Container services** > **Dynamic cache service** > **External cache groups**.

Name:

Specifies the external cache group name.

The external cache group name needs to match the ExternalCache property as defined in the servlet or JavaServer Pages (JSP) file cachespec.xml file.

When external caching is enabled, the cache matches pages with its Universal Resource Identifiers (URI) and pushes matching pages to the external cache. The entries can then be served from the external cache, instead of from the application server.

Type:

Specifies the external cache group type.

External cache group settings:

Use this page to configure sets of external caches that are controlled by WebSphere Application Server on web servers, such as IBM Edge Server and IBM HTTP Server.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Container services > Dynamic cache service** > **External cache groups** > *external_cache_group*.

Name:

Specifies the external cache group name.

The external cache group name must match the **ExternalCache** property as defined in the servlet or JavaServer Pages (JSPs) `cachespec.xml` file.

When external caching is enabled, the cache matches pages with its Universal Resource Identifiers (URIs) and pushes matching pages to the external cache. The entries can then be served from the external cache, instead of the application server. This ability creates a significant savings in performance.

External cache group member collection:

Use this page to define specific caches that are members of a cache group.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > *server_name* > **Container services > Dynamic cache service** > **External cache groups** > *external_cache_group* > **External cache group members**.

Address:

Specifies a configuration string that is used by the external cache adapter bean to connect to the external cache.

AdapterBeanName:

Specifies the adapter bean name.

Example adapter bean names that are supported in WebSphere Application Server are as follows:

Table 37. Adapter bean names.. Example adapter bean names

AFPA

Table 37. Adapter bean names. (continued). Example adapter bean names

AdapterBeanName: com.ibm.ws.cache.servlet.Afpa
Address: Port on which afpa listens
ESI
AdapterBeanName: com.ibm.websphere.servlet.cache.ESIInvalidatorServlet
IBM Web Traffic Express (WTE) (IBM Edge Server)
AdapterBeanName: com.ibm.websphere.edge.dynacache.WteAdapter
Address: hostname:port (host name and port on which WTE is listening)

External cache group member settings:

Use this page to define a single cache that is controlled by WebSphere Application Server.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers >** ***server_name*** **> Container services > Dynamic cache service > External cache groups >** ***external_cache_group*** **> External cache group members >** ***external_cache_group_member***.

Advanced Fast Path Architecture adapter bean name:

Specifies the adapter bean name.

- **Adapter bean name**: specifies the adapter bean name. For example, you can use a bean name such as `com.ibm.ws.cache.servlet.Afpa`.
- **Address:** specifies the port on which AFPA listens.

Edge Side Include (ESI):

Specifies the adapter bean name.

- **Adapter bean name**: specifies the adapter bean name. For example, you can use a bean name such as `com.ibm.websphere.servlet.cache.ESIInvalidatorServlet`.
- **Address:** local host

IBM Web Traffic Express (IBM WebSphere Edge Server):

Specifies the adapter bean name.

- **Adapter bean name**: specifies the adapter bean name. For example, you can use a bean name such as `com.ibm.websphere.edge.dynacache.WteAdapter`.
- **Address:** hostname:port (host name and port on which WTE is listening).

Configuring high-speed external caching through the web server:

IBM HTTP Server for Windows 2003 operating systems contains a high-speed cache referred to as the *Fast Response Cache Accelerator*, or *cache accelerator*. The Fast Response Cache Accelerator is available on Windows 2003 operating systems and AIX platforms. However, support to cache dynamic content is only available on Windows 2003 operating systems. You can enable cache accelerator to cache static and dynamic content.

Before you begin

About this task

Enable cache accelerator for caching static content by adding the following directives to the `httpd.conf` configuration file, located in the IBM HTTP Server `conf` directory:

- AfpaEnable
- AfpaCache on
- AfpaLogFile "*app_server_root*\IBMHttpServer\logs\afpalog" V-ECLF

To enable cache accelerator for caching dynamic content, such as servlets and JavaServer Pages (JSP) files, configure WebSphere Application Server and IBM HTTP Server for distributed platforms:

Procedure

1. Configure WebSphere Application Server to enable Fast Response Cache Accelerator.
   a. Configure an external cache group on the application server:
      1) Click **Servers > Server Types > WebSphere application servers > *server_name* > Container services > Dynamic cache service > External cache groups**.
      2) Click **New** on the External cache group administrative console page to define an external cache group named `afpa` for each application server that uses the cache accelerator.
      3) In the **External cache group** field, type `afpa` and apply the changes.
   b. Add a member to the group with an adapter bean name of `com.ibm.ws.cache.servlet.Afpa`.
      1) Click **Afpa > External cache group members**.
      2) Click **New** on the External cache group members administrative console page.
      3) In the AdapterBean name field, type `com.ibm.ws.cache.servlet.Afpa`.
      4) In the Address field, enter an unused port number.
   c. Add a cache policy in the `cachespec.xml` file for the servlet or JSP file you want to cache. Add the following property to the cache policy:

      ```
      <property name="ExternalCache">afpa</property>
      ```
2. Enable cache accelerator on IBM HTTP Server for distributed platforms:
   a. Add the following directives to the end of the `httpd.conf` file:
      - AfpaEnable
      - AfpaCache on
      - AfpaLogFile "*app_server_root*\IBMHttpServer\logs\afpalog" V-ECLF
      -

        depfeat: IBM HTTP Server 1.3.x - LoadModule afpaplugin_module *app_server_root*\bin\afpaplugin.dll
      - IBM HTTP Server 2.0 - LoadModule afpaplugin_20_module *app_server_root*\bin\afpaplugin_20.dll
      - AfpaPluginHost *WAS_Hostname:port*, where *WAS_Hostname* is the host name of the application server and *port* is the port you specified in the Address field while configuring the external cache group member

   The LoadModule directive loads the IBM HTTP Server plug-in that connects the Fast Response Cache Accelerator to the WebSphere Application Server fragment cache. If multiple IBM HTTP Servers are routing requests to a single application server, add the directives above to the `httpd.conf` file of each of these IBM HTTP Servers for distributed platforms.

Disabling template-based invalidations during JSP reloads

By setting the Java virtual machine (JVM) `com.ibm.ws.cache.CacheConfig.disableTemplateInvalidation` custom property to **true**, the template-based invalidations are disabled during JSP reloads.

About this task

To set any of these JVM custom properties, complete the following steps:

Procedure

1. In the administrative console, click **Servers > Application servers >*server_name* > Process definition > Java virtual machine > Custom properties > New**.

2. Enter `com.ibm.ws.cache.CacheConfig.disableTemplateInvalidation` in the Name field.
3. Enter **true** in the Value field.
4. Save the property and restart WebSphere Application Server.

Dynamic cache provider for the JPA 2.0 second level cache

Learn to use the WebSphere Application Server dynamic cache service as a Java Persistence API (JPA) second level (L2) cache provider. JPA 2.0 has standardized the L2 cache interface. WebSphere Application Server supports the JPA standard. The dynamic cache service plugs in as a level 2 cache provider to JPA. This topic describes how the L2 cache boosts the performance of your JPA application, the advantages of using DynaCache as L2 cache provider and how to configure and monitor the dynamic cache service for your JPA application in the WebSphere Application Server environment.

Note: A dynamic cache JPA second level (L2) cache provider for JPA 2.0 shares entity states across various persistence contexts, transactions and users. When caching is enabled, entities that are not found in the persistence context are loaded from the L2 cache. L2 caching avoids database access for currently-loaded entities. For more details about L2 caching, view the JPA 2.0 specification.

Attention: L2 caching increases the memory consumption of the application, therefore, it is important to limit the size of the L2 cache. There is also a possibility of stale data for updated objects in a clustered environment. Configure L2 caching for read-mostly, infrequently modified entities. L2 caches are not recommended for frequently and concurrently updated entities.

JPA utilizes several configurable L2 caches to maximize performance. The JPA data cache is a cache of persistent object data that operates at the EntityManagerFactory level. When enabled, the data cache is checked before making a trip to the datastore. Data is stored in the cache when objects are committed and when persistent objects are loaded from the datastore. In addition to the data cache, JPA defines service provider interfaces for a query cache.

The query cache stores the object IDs that are returned by query executions. When you run a query, JPA assembles a key that is based on the query properties and the parameters that are used at launch time and checks for a cached query result. If one is found, the object IDs in the cached result are looked up, and the resulting persistence-capable objects are returned. Otherwise, the query is launched against the database and the object IDs that are loaded by the query are placed into the cache. The object ID list is not cached until the list that is returned at query launch time is fully traversed.

The WebSphere Dynamic cache cache provider provides an excellent alternative to the default concurrent data and query cache providers on WebSphere Application Server, due to the value-added features that the dynamic cache service brings, such as its own feature set and the capabilities that are inherited from WebSphere Application Server. The dynamic cache provides the following advantages:

- Cluster distributed cache synchronization and replication through WebSphere Application Server data replication service (DRS) and high-availability (HA) services.
- Sophisticated and advanced DataCache and QueryCache monitoring, tuning and administration of the cache. The L2 cache will inherit the entire ecosystem of the available dynamic cache tooling.
- Native servant region z/OS support.
- Performs equally, if not better, to the default cache provider.

The JPA standard provides aliases as a mechanism to easily configure JPA plug-ins. The dynamic cache level 2 cache provider for JPA 2.0 is typically configured with the *dynacache* alias. You can use the *dynacache* alias for setting the DataCache and QueryCache properties of JPA. If the QueryCache property is set to use *dynacache*, then the DataCache property will also use the *dynacache* alias. You can configure the QueryCache and DataCache properties to use different types of cache providers. You can set the QueryCache property to default and set the DataCache to use dynamic cache, or reverse. It is ideal to use the WebSphere Application Server dynamic cache service for the DataCache, QueryCache, and the DataCacheManager. If the DataCache is set to **dynacache**, then the RemoteCommitProvider does

nothing because DRS and HAM are leveraged to do the cache synchronization. You cannot enable the QueryCache property separately from the DataCache. The QueryCache property benefits from the availability of the DataCache property, where the entity data is cached.

Configuring the JPA dynamic cache L2 provider (basic method)

Enable the dynamic cache service as the level 2 cache provider for JPA 2.0 by setting some or all of the following properties to **dynacache**: OpenJPA openjpa.QueryCache, openjpa.DataCache, and the openjpa.DataCacheManager. This default configuration is a suitable default for most environments.

```
<property name="openjpa.DataCache" value="dynacache(CacheSize=1000)"/>
<property name="openjpa.QueryCache" value="dynacache"/>
<property name="openjpa.DataCacheManager" value="dynacache"/>
```

Table 38. Property names and aliases.. Fully-qualified property names and aliases for the dynamic cache.

Dynamic cache name	Property name	Alias
DataCache cache provider	com.ibm.ws.cache.openjpa. DynacacheDataCache	dynacache
QueryCache cache provider	com.ibm.ws.cache.openjpa. DynacacheQueryCache	dynacache
RemoteCommitProvider	com.ibm.ws.cache.openjpa. NoOpRemoteCommitProvider	none
DataCacheManager	com.ibm.ws.cache.openjpa. DynacacheDataCacheManager	dynacache

Configuring the JPA dynamic cache L2 provider (advanced method)

Configure the dynamic cache instance for the JPA 2.0 Level 2 data or query cache with additional or advanced configuration properties in the persistent unit (advanced method). Enable the dynamic cache service as the level 2 cache provider for JPA 2.0 by setting some or all of the following properties to **dynacache**: OpenJPA openjpa.QueryCache, openjpa.DataCache, and openjpa.DataCacheManager properties.

```
<property name="openjpa.DataCache" value="dynacache(CacheName="myJPACache",
                                                    CacheSize=1000,
                                                    EnableDiskOffload=true,
                                                    DiskCacheSizeInGB=4,
                                                    DiskOffloadLocation=c:\temp,
                                                    EnableCacheReplication=true)"/>
```

Table 39. Dynamic cache property names and values. Properties of the dynamic cache cache instance that can be configured in the persistent unit.

Dynamic cache custom properties	Default values
CacheName	default
CacheSize	2000
EnableDiskOffload	false
EnableCacheReplication	false
DiskCacheSizeInGB	not applicable
DiskOffloadLocation	not applicable
ReplicationType	1 , 2 or 4. 1 means NOT_SHARED, 2 is PUSH and 4 is PUSH_PULL

Important: The properties in the table are not mandatory. If not specified, the dynamic cache service assumes default values that are suitable for the majority of users.

The following example showcases the dynamic cache mbean operations in action to introspect the L2 cache:

```
wsadmin>set mbean [$AdminControl queryNames type=DynaCache,*]

wsadmin> $AdminControl invoke $mbean getCacheIDsInMemory {default \S}
66
71
10
A5614-67
```

```
wsadmin> $AdminControl invoke $mbean getJPACacheStatistics {OpenBooks openbooks.war OpenBooks default}
HIT_COUNT=0
TOTAL_HIT_COUNT=0
READ_COUNT=5
TOTAL_READ_COUNT=5
WRITE_COUNT=4
TOTAL_WRITE_COUNT=4

wsadmin> $AdminControl invoke $mbean getJPACacheStatistics {OpenBooks openbooks.war OpenBooks default openbook.domain.Customer}
HIT_COUNT=0
TOTAL_HIT_COUNT=0
READ_COUNT=0
TOTAL_READ_COUNT=0
WRITE_COUNT=1
TOTAL_WRITE_COUNT=1
```

You can also use the Extended Cache Monitor to view the contents of the cache ID and key values that are placed in the cache by the JPA runtime.

Using the dynamic cache L2 cache provider in a clustered environment

Customers can define a cache-instance in the customary way in WebSphere Application Server and then reference the name in the CacheName property of the JPA DataCache property value. You can define an object cache instance using the cacheinstances.properties file, DistributedMapFactory, or the administrative console.

All properties are optional and if they are not specified, defaults properties are assumed. If the cache instance does not exist, the dynamic cache service creates the cache instance. Advanced configuration for a QueryCache Dynacache instance is completed in a similar manner.

The JPA data cache operates in both single-JVM and multi-JVM environments. Multi-JVM caching is achieved through the use of the Data Replication Service (DRS) in a clustered WebSphere Application Server, Network Deployment environment.

For entities and queries in the dynamic cache DataCache and QueryCache instances that are replicated across servers in a WebSphere Application Server, Network Deployment environment using the Data Replication Service, configure a replication domain and associate the replication domain with the cache instance. The persistent unit must also have the openjpa.RemoteCommitProvider openJPA property set to **none**.

Replicate an OpenJPA L2 dynamic cache instance, as follows:

1. Create a replication domain in the administrative console and associate the replication domain with the baseCache cache instance on the dynamic cache service panel on all application servers that must share the distributed cache. The dynamic cache service uses this replication domain to replicate data across servers in the replication domain.
2. Configure the cache instance by setting the enableCacheReplication property to true when defining the configuration of the cache instance. If you do not specify the replicationType property, the cache instance is configured by default in the NOT_SHARED sharing mode, where only invalidations are propagated. You can configure the cache instance in the NOT_SHARED, PUSH and PUSH_PULL sharing types. Refer to the cache replication topic to learn more about this topic.
3. Set the `com.ibm.ws.cache.CacheConfig.createCacheAtServerStartup` JVM custom property to **true** on all the application servers in the replication domain. This custom property helps the JPA cache instances on servers bootstrap earlier and faster.
4. Create a shared library to make the entity classes available in the classpath of the application server by defining a shared library and associating it with the server classloader. This step is necessary for the dynamic cache service and DRS to deserialize the replicated entity objects. Refer to the Shared library collection, Managing shared libraries, and Associating shared libraries with servers topics for more information about shared libraries.
5. Set the openjpa.RemoteCommitProvider to **none**.

You might also use other RemoteCommitProvider implementations that are included with JPA 2.0 with dynamic cache, specifically the following implementations:

- org.apache.openjpa.event.SingleJVMRemoteCommitProvider (configured with the "sjvm" alias)
- org.apache.openjpa.event.TCPRemoteCommitProvider
- org.apache.openjpa.event.JMSRemoteCommitProvider

Read more about remote and offline operation in the JPA documentation.

Attention: Configure dynamic cache with DRS for replication of JPA data and QueryCache objects and using the NoOp RemoteCommitProvider property in a distributed or clustered environment.

Troubleshooting the JPA L2 cache

Look for the following messages in the log file when using the dynamic cache service as a JPA cache provider:

```
# Significant dynamic cache OpenJPA messages in the SystemOut.log file
DYNA1081I: OpenJPA L2 DataCache Dynacache instance \"{0}\" created or retrieved successfully for persistent unit \"{1}\".

# Applicable only if QueryCache is enabled
DYNA1080I: OpenJPA L2 QueryCache Dynacache instance \"{0}\" created or retrieved successfully for persistent unit \"{1}\".
```

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Chapter 9. Administering EJB applications

This page provides a starting point for finding information about enterprise beans.

Based on the Enterprise JavaBeans (EJB) specification, enterprise beans are Java components that typically implement the business logic of Java 2 Platform, Enterprise Edition (J2EE) applications as well as access data.

Deploying EJB 3.x enterprise beans

EJB module settings

Use this page to configure and manage a specific deployed EJB module.

Note: You cannot start or stop an individual EJB module for modification. You must start or stop the appropriate application entirely.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > **application_name** > **Manage Modules** > **module_name**.

Attention: If an application is running, changing an application setting causes the application to restart. On stand-alone servers, the application restarts after you save the change. On multiple-server products, the application restarts after you save the change and files synchronize on the node where the application is installed. To control when synchronization occurs on multiple-server products, deselect **Synchronize changes with nodes** on the Console preferences page.

URI

Specifies location of the module relative to the root of the application EAR file. The URI must match the URI of a ModuleRef URI in the deployment descriptor of the deployed application (EAR).

Alternate deployment descriptor

Specifies an alternate deployment descriptor for the module as defined in the application deployment descriptor according to the Java Platform, Enterprise Edition (Java EE) specification.

Starting weight

Specifies the order in which modules are started when the server starts. The module with the lowest starting weight is started first.

If the application deployment descriptor specifies the `<initialize-in-order>true</initialize-in-order>` element, the default starting weights reflect the order that is specified in the deployment descriptor. Otherwise, the defaults are determined based on module type (RAR modules start before EJB modules, which start before web modules).

Data type	Integer
Default	5000
Range	Greater than 0

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 40. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/`*`profile_name`* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express` directory.

The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Deploying EJB modules

When you deploy an Enterprise JavaBeans (EJB) module, you install that module on a server that has been configured to support deployed modules.

Before you begin

Assemble one or more EJB modules, assemble one or more web modules, and assemble them into a Java EE application.

For an overview about the changes to the EJB deployment model for EJB 3.x, see the topic EJB 3.x deployment overview.

Procedure

1. Prepare the deployment environment. See the topic Preparing to host applications.
2. Update the configuration for each EJB module as needed for the deployment environment.
3. Required: If a module has dependencies on Java 5-specific extensions, such as generics, annotations, and so on, then you must run the EJBDeplolycommand-line tool separately and before installing the module or application containing it. This is because the administrative console and the wsadmin command-line tool do not allow for specifying the ejbdeploy -complianceLevel 5.0 option.

   It is only necessary to run the EJBDeploy tool for EJB 2.1 modules containing entity beans.
4. Address potential interoperability issues.

   There can be unexpected results if a WebSphere stack product, or another product, that runs on a version of Application Server that does not support EJB 3.x attempts to remotely invoke a method on an EJB 3.x compliant enterprise bean on a separate server that is running a version Application Server that supports EJB 3.x. If these products attempt to invoke a method through the enterprise bean's EJB 3.x remote business interface, they might encounter exceptions that were introduced in EJB 3.x that will be pushed back to the environment that is not EJB 3.x compliant.

   This scenario could also be an issue for an administrator of an environment that includes a combination of stack products that contain a mixture of EJB 3.x compliant and non-compliant instances of Application Server.

   The following is a list of the exception classes that have been introduced in EJB 3.0:

   - javax.ejb.ConcurrentAccessException
   - javax.ejb.EJBAccessException
   - javax.ejb.EJBTransactionRequiredException
   - javax.ejb.EJBTransactionRolledbackException
   - javax.ejb.NoSuchEJBException

   a. Ensure that Application Server is updated to 7.0.0.3.
   b. Manually copy the <*app_server_root*>/runtimes/ejb3exceptions.jar file from Application Server to a directory on each of the stack products installations, or other product installations, that you will use as the EJB 3.x client.

c. Ensure that the directory that contains the ejb3exceptions.jar file is in the class path. One possible location for the JAR file that would satisfy this requirement is the <*app_server_root*>/lib directory on a server that is not EJB 3.x compliant.

   Note: Just like the EJB thin client jars, if an update becomes available, users must copy the ejb3exceptions.jar file again after installing the version of the WebSphere Application Server containing the updated version.

5. Deploy the application. See the topic Deploying and administering enterprise applications.

What to do next

If you specify that the EJBDeploy tool be run during application installation and the installation fails with a NameNotFoundException message, ensure that the input Java archive (JAR) or enterprise archive (EAR) file does not contain source files. Either remove the source files or include all dependent classes and resource files on the class path. If there are source files in the input JAR or EAR file, the EJB deployment tools runs a rebuild before generating the deployment code.

If the module deploys successfully, test and debug the module. See the topic Diagnosing problems (using diagnosis tools).

EJB 3.0 and EJB 3.1 deployment overview

Learn about the Enterprise JavaBeans (EJB) 3.0 and 3.1 deployment model, including *Just-In-Time* (JIT) deployment.

All Java Enterprise Edition (Java EE) application server products have some form of EJB deployment phase in which your application is customized to run in that particular implementation of the application server. Typically, this is accomplished by a deployment tool that is specific to the application server and generates code to bridge your EJB interface and implementation code to the application server's implementation for an EJB container. Some application server products' deployment tools alter the bytecodes of your application classes, rather than generating code, but the end result is similar.

Application Server bridges your EJB interface with its implementation by generating code that encapsulates your EJB implementation classes, connecting them to Application Server's EJB container. This enables the EJB container to host your enterprise beans and provide services to them. If one or more of your enterprise beans has remote interfaces defined, Application Server generates additional code to provide the remote function.

For more information about packaging your EJB module, see the topic that covers the EJB 3.x module packaging overview.

EJBDeploy Tool

Historically, EJB deployment in the Application Server product has been performed by the EJBDeploy tool, which is included with WebSphere Application Server and packaged with the development tools for the WebSphere products.

The EJBDeploy tool introspects the external interfaces for your enterprise beans, generates the wrapper code as .java files, and compiles the code using the javac compiler to produce .class files that are packaged in your EJB module with your application code. The EJBDeploy tool also runs the rmic tool against the remote EJB interfaces in the application, producing additional *stub* and *tie* class files that interact with the Remote Method Invocation over Internet Inter-ORB Protocol (RMI-IIOP) Object Request Broker (ORB), providing remote object support.

For modules previous to EJB 3.0, you ran the EJBDeploy tool when you installed the application on Application Server or before you installed the application from the command-line tool or a development tool.

Just-In-Time (JIT) deployment

EJB 3.0 support in Application Server introduced a new feature called JIT deployment.

With JIT deployment, the EJB container dynamically generates the wrapper, stub, and tie classes in-memory when the application is running. Additionally, the web container and application client containers dynamically generate the stub class that is required for remote EJB invocations.

Effectively, this means that you do not need to process EJB 3.0 or 3.1 modules, web modules that invoke EJB 3.0 or 3.1 beans, or client modules that invoke EJB 3.0 or 3.1 beans through the EJBDeploy tool before you run them in Application Server.

createEJBStubs tool

In most cases the Just-In-Time deployment feature can dynamically generate the RMI-IIOP stub classes that are required for invocation of remote EJB interfaces. There are some instances in which these stub classes are not dynamically generated. For EJB 3.0 or 3.1 clients that are not running inside an EJB 3.x enabled web container, EJB container, or client container, you must generate the stub classes with the createEJBStubs tool and ensure that the generated stubs are available in the client environment's class path. Typically, you would accomplish this by copying the generated stubs to the location where the client's business interface class resides.

The createEJBStubs tool must be used to generate client-side stubs for the following environments:

- "Bare" Java Standard Edition (SE) clients, where a Java SE Java Virtual Machine (JVM) is the client environment.
- Containers in Application Server environments prior to Version 7 that do not have the Feature Pack for EJB 3.0 applied.
- Environments that are not WebSphere Application Server environments.

Interoperability

There can be unexpected results if a WebSphere stack product, or another product, that runs on a version of Application Server that does not support EJB 3.0 or 3.1 attempts to remotely invoke a method on an EJB 3.x compliant enterprise bean on a separate server that is running a version Application Server that supports EJB 3.0 or 3.1. If these products attempt to invoke a method through the enterprise bean's EJB 3.x remote business interface, they might encounter exceptions that were introduced in EJB 3.0 that will be pushed back to the environment that is not EJB 3.x compliant.

This scenario could also be an issue for an administrator of an environment that includes a combination of stack products that contain a mixture of EJB 3.x compliant and non-compliant instances of Application Server.

The following is a list of the exception classes introduced in EJB 3.0:

- javax.ejb.ConcurrentAccessException
- javax.ejb.EJBAccessException
- javax.ejb.EJBTransactionRequiredException
- javax.ejb.EJBTransactionRolledbackException
- javax.ejb.NoSuchEJBException

Refer to the EJB module deployment step to address potential interoperability issues.

EJB 2.x Modules

EJB 2.x modules that have been converted to be EJB 3.0 or EJB 3.1 modules should have all WebSphere Application Server generated files (including stub and tie classes) removed prior to EJB deployment in the Application Server product.

EJBDEPLOY relationships – troubleshooting tips

Use this information to troubleshoot information for EJBDEPLOY problems.

The converter that is defined for the primary key is not invoked on its foreign key value

The mapping for primary key fields to database columns may use a converter to transform the key values. If a container-managed persistence (CMP) bean uses a converter to map its primary key, and that bean has a relationship where the bean at the other end holds a foreign key, the mapping for the foreign key will not use the converter.

The following errors might occur, indicating that the converter defined for the primary key is not invoked on its foreign key value. During the run of the ejbDeploy command , you receive the following message:

```
No type mapping defined for Java datatype1 to Database datatype2
```

During run time, the application does not find the CMP bean at the other end of the relationship.

To work around this limitation, define your own foreign key in the database table, and create a mapping that uses the same converter as defined for the primary key on the enterprise beans at the other end of its relationship.

Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
: The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
: The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
: The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
: The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 41. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit
64–bit IBM Technology for Java	/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit

plugins_profile_root
The default Web Server Plug-ins profile root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the /QIBM/ProdData/WebSphere/Plugins/V8/webserver directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the /QIBM/UserData/WebSphere/Plugins/V8/webserver directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to .exe, .dll, .so objects) for the installed product. The product library name is QWAS8*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is QWAS8A. The *app_server_root*/properties/product.properties file contains the value for the product library of the installation, was.install.library, and is located under the *app_server_root* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the /QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/*profile_name* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is QWAS8. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server - Express is the /QIBM/UserData/WebSphere/AppServer/V8/Express directory.

The profiles and profileRegistry subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Administering entity beans

Enterprise beans back up and recovery best practices

The following items should be considered for backup when using enterprise beans.

Database data

Enterprise beans often use a database for back-end persistence. Container-managed persistence (CMP) entity beans always use a database for back-end persistence. This data should be backed up the same as any of your business data.

The collection for container-managed entity beans persistence is determined by either the schema name specified during deployment, or the schema specified on the data source associated with the enterprise bean. Any persistent store used by session and bean-managed beans is defined by the bean implementation. For database tables, you can choose to save the entire collection or an individual table as shown with the following commands, respectively:

```
SAVLIB LIB(EJB) DEV(*SAVF) SAVF(WSALIB/WSASAVF)
SAVOBJ OBJ(MYBEANTBL) LIB(EJB) DEV(*SAVF) OBJTYPE(*FILE) SAVF(WSALIB/WSASAVF)
```

It might be possible to save database objects while the product is active, if the save operation can obtain a snapshot of the data store. You may have to shut down if a snapshot cannot be obtained. This occurs if there are requests that obtain locks or have open transactions against the database being saved.

Enterprise JavaBeans (EJB) source code, class files, and deployed code

When you deploy enterprise beans, a WebSphere Application Server-specific implementation of the enterprise beans is generated. Save these deployed Java(TM) Archive (JAR) files to avoid redeploying, and to preserve any binding information that was specified during the application installation. The JAR files, application code and configuration, such as bindings, are located by default in the *profile_root*/installedApps directory. By saving this directory, you save your installed applications, including HTML, servlets, JavaServer Pages(TM) (JSP(TM)) files, and enterprise beans. Normally, each application is located in a separate subdirectory, so you can choose to save all applications or a subset.

Note: The commands below have been wrapped for display purposes. Enter each as a single command.

This command saves all installed applications:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('/profile_root/installedApps'))
```

This command saves the sampleApp application only:

```
SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
 OBJ(('/profile_root/installedApps/cellName/sampleApp.ear'))
```

If you have located utility or general purpose classes in other directories, such as /*profile_root*/lib/app or /*profile_root*/lib/ext, be sure to include those locations in your backup plan as well.

Administrative configuration

For more information, see the topic Introduction: Administrative configuration data.

Managing EJB containers

Each application server can have a single Enterprise JavaBeans (EJB) container; one is created automatically for you when the application server is created. The following steps are to be performed only as needed to improve performance after the EJB application has been deployed.

Procedure

1. Adjust EJB container settings.
2. Adjust EJB cache settings.

What to do next

If adjustments do not improve performance, consider adjusting access intent policies for entity beans, reassembling the module, and redeploying the module in the application.

EJB containers

An Enterprise JavaBeans (EJB) container provides a run-time environment for enterprise beans within the application server. The container handles all aspects of an enterprise bean's operation within the application server and acts as an intermediary between the user-written business logic within the bean and the rest of the application server environment.

One or more EJB modules, each containing one or more enterprise beans, can be installed in a single container.

The EJB container provides many services to the enterprise bean, including the following:

- Beginning, committing, and rolling back transactions as necessary.
- Maintaining pools of enterprise bean instances ready for incoming requests and moving these instances between the inactive pools and an active state, ensuring that threading conditions within the bean are satisfied.
- Most importantly, automatically synchronizing data in an entity bean's instance variables with corresponding data items stored in persistent storage.

By dynamically maintaining a set of active bean instances and synchronizing bean state with persistent storage when beans are moved into and out of active state, the container makes it possible for an application to manage many more bean instances than could otherwise simultaneously be held in the application server's memory. In this respect, an EJB container provides services similar to virtual memory within an operating system.

WebSphere Application Server provides significant flexibility in the management of database data with entity beans. The Entity EJBs Activate at and Load at configuration settings specify how and when to load and cache data from the corresponding database row data of an enterprise bean. These configuration settings provide the capability to specify enterprise bean caching Options A, B or C, as specified in the EJB 1.1 specifications. You can configure these settings with assembly tools. To read more about how to use the assembly tools see the assembly tool information center.

Option A provides maximum enterprise bean performance by caching database data outside of the transaction scope. Generally, Option A is only applicable where the EJB container has exclusive access to the given database. Otherwise, data integrity is compromised. Option B provides more aggressive caching of Entity EJB object instances, which can result in improved performance over Option C, but also results in greater memory usage. Option C is the most common real-world configuration for Entity EJBs and is the default setting.

The `Activate at` setting specifies the point at which an enterprise bean is activated and placed in the cache. Removal from the cache and passivation are also governed by this setting. Valid values are `Once` and `Transaction`. The `Once` setting indicates that the bean is activated when it is first accessed in the server process, and passivated (and removed from the cache) at the discretion of the container, for example when the cache becomes full. The `Transaction` setting indicates that the bean is activated at the start of a transaction and passivated (and removed from the cache) at the end of the transaction. The default value is `Transaction`.

The `Load at` setting specifies when the bean loads its state from the database. The value of this property implies whether the container has exclusive or shared access to the database. Valid values are `Activation` and `Transaction`. Activation indicates the bean is loaded when it is activated and implies that the container has exclusive access to the database. Transaction indicates that the bean is loaded at the start of a transaction and implies that the container has shared access to the database. The default is `Transaction`. The settings of the `Activate at` and `Load at` properties govern which commit options are used. For Option A (exclusive database access), use `Activate at = Once` and `Load at = Activation`. This option reduces database input/output by avoiding calls to the ejbLoad function, but serializes all

transactions accessing the bean instance. Option A can increase memory usage by maintaining more objects in the cache, but can provide better response time if bean instances are not generally accessed concurrently by multiple transactions.

Important: When using WebSphere WebSphere Application Server, Network Deployment and workload management is enabled, Option A cannot be used.

You must use settings that result in the use of Options B or C. For Option B (shared database access), use `Activate at = Once` and `Load at = Transaction`. Option B can increase memory usage by maintaining more objects in the cache. However, because each transaction creates its own copy of an object, there can be multiple copies of an instance in memory at any given time (one per transaction), requiring the database be accessed at each transaction. If an enterprise bean contains a significant number of calls to the ejbActivate function, using Option B can be beneficial because the required object is already in the cache. Otherwise, this option does not provide significant benefit over Option A. For Option C (shared database access), use `Activate at = Transaction` and `Load at = Transaction`. This option can reduce memory usage by maintaining fewer objects in the cache. However, there can be multiple copies of an instance in memory at any given time (one per transaction). This option can reduce transaction contention for enterprise bean instances that are accessed concurrently but not updated.

This product supports the cloning of stateful session bean home objects among multiple application servers. However, it does not support the cloning of a specific instance of a stateful session bean. Each instance of a particular stateful session bean can exist in just one application server and can be accessed only by directing requests to that particular application server. State information for a stateful session bean cannot be maintained across multiple members of a server cluster. However, enabling stateful session bean failover and configuring the EJB container to use memory-to-memory replication does enable stateful session bean failover to be replicated to other servers in the cluster so that failover can occur to the backup server if the primary server for a stateful session bean stops for some reason. For more information about stateful session bean failover, see Stateful session bean failover for the EJB container.

By default, an EJB container runs in the **quick start** mode. The EJB container startup logic delays the loading and processing of all EJB types *except* Message Driven Beans, because message driven beans must exist before messages are posted for them; Startup Beans, which must be processed when the server starts; and EJB types that you specify to initialize when the server starts. .

All other EJB initialization is delayed until the first use of the EJB type. When using local interfaces, the first use is when you perform an InitialContext.lookup method for the type. For remote interfaces, it is when you call the first method on an EJB or its Home.

EJB container settings

Use this page to configure and manage the EJB container of this application server.

To view this administrative console page, click **Servers** > **WebSphere application servers** > **server** > **EJB Container Settings** > **EJB container**.

Passivation directory

Specifies the directory into which the container saves the persistent state of passivated stateful session beans. This directory must already exist. It is not automatically created.

Stateful session beans with an activation policy of TRANSACTION are passivated at the end of the transaction in which they are enlisted, and stateful session beans with an activation policy of ONCE (default) are passivated when the number of active bean instances becomes greater than the cache size specified in the container configuration. When a stateful bean is passivated, the container serializes the bean instance to a file in the passivation directory and discards the instance from the bean cache. If, at a later time, a request arrives for the passivated bean instance, the container retrieves it from the

passivation directory, deserializes it, returns it to the cache, and dispatches the request to it. If any step fails (for example, if the bean instance is no longer in the passivation directory), the method invocation fails.

Inactive pool cleanup interval

Specifies the interval at which the container examines the pools of available bean instances to determine if some instances can be deleted to reduce memory usage. This setting is for all bean pools. **Attention:** Stateful session beans are NOT pooled, so this applies to stateless session and entity bean pools.

Data type	Integer
Default	30000
Units	Milliseconds
Range	0 to 2 147 483 647

Default data source JNDI name

Specifies the JNDI name of a data source to use if no data source is specified during application deployment. This setting is not applicable for EJB 2.x-compliant CMP beans.

Servlets and enterprise beans use *data sources* to obtain these connections. When configuring a container, you can specify a default data source for the container. This data source becomes the default data source used by any entity beans installed in the container that use container-managed persistence (CMP).

The default data source for a container is secure. When specifying it, you must provide a user ID and password for accessing the data source.

Specifying a default data source is optional if each CMP entity bean in the container has a data source specified in its configuration. If a default data source is not specified and a CMP entity bean is installed in the container without specifying a data source for that bean, applications cannot use that CMP entity bean.

Enable stateful session bean failover using memory-to-memory replication

Specifies that failover is enabled for *all* stateful session beans installed in this EJB container.

This checkbox is disabled until you define a replication domain. This selection has a hyperlink to help you configure the replication settings. If no replication domains are configured, the link takes you to a panel where you can create one. If at least one domain is configured, the link takes you to a panel where you can select the replication settings to be used by the EJB container.

Data type	Checkbox
Default	Unselected
Range	Selected or unselected.

Initial state

Specifies the execution state requested when the server first starts.

Data type	String
Default	Started
Range	Valid values are `Started` and `Stopped`

EJB container system properties

In addition to the settings that are accessible from the administrative console, you can set EJB system properties using command-line scripting.

You can use the properties page to define the following EJB container system properties:

- “com.ibm.websphere.ejbcontainer.allowEarlyInsert”
- “com.ibm.websphere.ejbcontainer.checkEJBApplicationConfiguration”
- “com.ibm.websphere.ejbcontainer.declaredUncheckedAreSystemExceptions” on page 395
- “com.ibm.websphere.ejbcontainer.defaultStatefulSessionTimeout” on page 395
- “ com.ibm.websphere.ejbcontainer.EE5Compatibility ” on page 395
- “com.ibm.websphere.ejbcontainer.limitSetRollbackOnlyBehaviorToInstanceFor” on page 396
- “com.ibm.websphere.ejbcontainer.poolSize” on page 396

com.ibm.websphere.ejbcontainer.allowEarlyInsert

This property is applicable to container managed persistence (CMP) 1.1 beans only. By default, the EJB container creates the entity bean representation in the database only after the method, ejbPostCreate(...), is called.

Note: CMP beans are not supported in EJB 3.x modules.

Some applications might rely on method, ejbCreate(...), to have created the entity bean in the database. For such a requirement, setting the JVM property, com.ibm.websphere.ejbcontainer.allowEarlyInsert, to `true` overrides the default behavior.

com.ibm.websphere.ejbcontainer.checkEJBApplicationConfiguration

Specifies a server-wide setting that indicates the container should complete additional application configuration validation to ensure the application is consistent with the Java Platform, Enterprise Edition (Java EE) specification.

Note: You can also specify this property as an application custom property.

This property is intended for use during development of an application to assist in identifying improper configurations, which might result in unexpected behavior. For example, applying the javax.ejb.Asynchronous annotation to an interface is not supported by the specification and is typically ignored. When this property is enabled, an error is logged, and an exception occurs when the bean is processed. This failure is useful during development to understand why the methods are not working asynchronously.

Additional configuration validation is completed and might result in multiple configuration warnings and errors being logged when this property is enabled. Typically, this additional validation is for scenarios that incur extra overhead to run, which is unnecessary for stable applications on a production server. For minor deviations from the specification, only warnings are logged. For more significant issues, an error is logged, and the application cannot run until the error is corrected.

Note: When the com.ibm.websphere.ejbcontainer.checkEJBApplicationConfiguration property is enabled, some of the issues that were reported as warnings are displayed as errors, which prevents the application from starting.

Note: The embeddable container and servers configured for development mode perform the additional validation that is associated with this property; however, all identified issues are reported as warnings, rather than errors, unless this property is specifically enabled.

com.ibm.websphere.ejbcontainer.declaredUncheckedAreSystemExceptions

This property enables you to indicate whether exceptions that are declared on the throws clause of an EJB method should are treated as application exceptions or as system runtime exceptions. When this property is set to `true`, these exceptions are treated like system runtime exceptions, and causes an EJBException to be issued on the client side.

If this property is not specified, or if this property is set to `false`, exceptions that are declared on the throws clause of an EJB method are treated as application exceptions.

The default value for this property is false.

transition: In Version 7, the default value for this property is true.

com.ibm.websphere.ejbcontainer.defaultStatefulSessionTimeout

Specifies a server-wide timeout for stateful session beans, which indicates how long a stateful session bean is retained by the server.

Note: The property applies only to EJB 3.1 modules and later.

This is a system property that you can add directly to the server.xml file or as a generic JVM argument using the administrative console.

The property is specified in minutes, the only valid unit. The default value is 10 minutes. A value of zero specifies that the server uses the default value of 10 minutes. A negative value is not valid. Any zero or greater value is valid. If a non-valid value is specified, a warning is issued to SystemOut, and the default value is used.

The stateful session bean timeout duration can be specified on a *per-bean* basis using annotations or xml. If a timeout duration is explicitly specified for a particular bean, this takes precedence over any server-wide timeout setting.

If no bean-specific timeout duration exists for a particular bean, then the server-wide timeout setting is applied to that bean.

If no bean-specific timeout duration exists for a particular bean, and no server-wide timeout setting exists, then the default timeout setting is applied to that bean.

com.ibm.websphere.ejbcontainer.EE5Compatibility

Specifies a server-wide setting that indicates the EJB container provides default behaviors that are consistent with the Java Enterprise Edition (Java EE) 5.0 specification.

This is a system property that you can add directly to the server.xml file or as a generic JVM argument using the administrative console.

The Java EE specification includes improvements to the EJB programming model that have resulted in minor changes to some default behaviors. In general, these changes provide more intuitive or more reliable behavior. However, if an application has been written to rely on one or more of the Java EE 5.0 behaviors, this system property might be set to revert the EJB container back to the Java EE 5.0 default behaviors.

Setting the property to true overrides the following behaviors:

- @ApplicationException annotations are not inherited. Starting with the Java EE 6.0 specification, the default behavior for the @ApplicationException annotation was changed, indicating that the annotation

is inherited by subclass exception classes. When this system property is specified, the @ApplicationException annotation is not inherited by subclass exception classes. Alternatively, you can change the @ApplicationException declaration to specify 'inherited=false'.

- Concurrent access to stateful session beans is prohibited, and results in the exception, javax.ejb.ConcurrentAccessException. Starting with the Java EE 6.0 specification, the default behavior for stateful session concurrency was changed to allow concurrent access, though each concurrent request is serialized by the container, blocking access to the bean instance indefinitely until the instance lock may be obtained. When this system property is specified, all stateful session beans that do not have an explicit access-timeout value specified assume the default access-timeout value of 0 (the Java EE 5.0 default). Alternatively, you can modify the stateful session bean to define an access-timeout value of 0.
- The Java type, Class, and any subclass of Enum are treated as resource environment references instead of simple environment entries. Starting with the Java EE 6.0 specification, the Java type, Class, and any subclass of Enum were added to the set of supported simple environment entry types. In prior versions of Java EE, these types would have been treated as resource environment references and a binding would have been required in the `ibm-ejb-jar-bnd.xml` file or `ibm-web-bnd.xml` file. Now that these data types are supported as simple environment entries, a platform-specific binding is no longer required. Instead, you can specify the value directly in the deployment descriptor. When this system property is specified, applications that use the javax.annotation.Resource annotation for the Java type, Class, or any subclass of Enum is treated as resource environment references, and the referenced value will be obtained using the binding file information. Installing the application is not effected by this property, and therefore, binding information is not entered during installation. Instead, you must manually enter the binding information in the binding file.

com.ibm.websphere.ejbcontainer.extendSetRollbackOnlyBehaviorToInstanceFor

This property allows the user to specify application names in which they want to have the EJBs in their EJB 3.x modules demonstrate the pre-EJB 3.0 setRollbackOnly behavior.

The pre-EJB 3.0 setRollbackOnly behavior is described in “Changing applications to WebSphere "version specific" setRollbackOnly behavior” on page 398.

com.ibm.websphere.ejbcontainer.limitSetRollbackOnlyBehaviorToInstanceFor

This property allows the user to specify application names in which they want to have the EJBs in their EJB 3.x modules demonstrate the EJB 3.x setRollbackOnly behavior.

The EJB 3.x setRollbackOnly behavior is described in “Changing applications to WebSphere "version specific" setRollbackOnly behavior” on page 398.

com.ibm.websphere.ejbcontainer.poolSize

Specifies the size of the pool for the specified bean type. This property applies to stateless, message-driven, and entity beans. If you do not specify a default value, the container default value, 50 and 500, are used.

Set the pool size for a given entity bean as:

beantype=[*H*]*min*,[*H*]*max* [:*beantype*=[*H*]*min*,[*H*]*max*...]

The *beantype* element is the Java EE name of the bean, formed by concatenating the application name, the **#** character, the module name, the **#** character, and the name of the bean, that is, the string assigned to the `<ejb-name>` field in the deployment descriptor of the bean. The *min* and *max* elements are the minimum and maximum pool sizes for that bean type. Do not specify the square brackets shown in the previous prototype; they denote optional additional bean types that you can specify after the first bean type. Each bean type specification is delimited by a colon (:).

Use an asterisk (*) as the value of *beantype* to indicate that all bean types are to use those values unless overridden by an exact bean-type specification somewhere else in the string; for example:

```
*=30,100
```

To specify a default value, omit either the *min* or *max* value but retain the comma (,) between the two values; for example:

Note: The following example displayed on multiple lines for publication purposes.

```
SMApp#PerfModule#TunerBean=54,
   :SMApp#SMModule#TypeBean=100,200
```

You can specify the bean types in any order within the string.

You can designate the maximum configured EJB pool size as a *hard limit* by inserting the character, *H,* directly in front of the *max* value. Without the *H character*, the maximum value indicates how many EJB instances can be pooled and does not limit the number of EJB instances that can be created or in use. Inserting the *H* character before the max value indicates a hard limit, and the EJB container blocks creation of more instances when that limit is reached. Further threads must wait until an instance becomes available or until the transaction times out.

You can designate the minimum configured EJB pool size as a *hard limit* by inserting the character, *H,* directly in front of the *min* value. Without the *H* character, the minimum value indicates how many EJB instances are maintained in the pool when the EJB type is not actively in use, but does not preload the pool when the application is started. Typically, the minimum pool size is not reached until the minimum number of EJB instances has been accessed concurrently by the application. Inserting the *H* character before the min value indicates a hard limit, and the EJB container preloads the pool with the minimum number of EJB instances when the application is started.

For example, if you want to indicate that no more than 200 EJB instances are created, and that additional requests must wait for an instance and might time out while waiting, then enter:

```
SMApp#SMModule#TypeBean=100,H200
```

If you want to indicate that the EJB container preloads the pool with a minimum of 100 EJB instances when the application is started, then enter:

```
SMApp#SMModule#TypeBean=H100,200
```

Note: The hard limit indicator is only available to EJB Version 2.0 or later stateless session beans.

Changing enterprise bean types to initialize at application start time using the administrative console

All Enterprise JavaBeans (EJB) types within a server can be forced to initialize at application start time by setting a system property within the administrative console.

About this task

If the value of this property is set to **true**, then all beans within the server are initialized at each application's start time.

However, by default, the product's EJB container delays the initialization (loading of classes and processing of deployment descriptor metadata) of most EJB types until they are needed during run time. This delay helps to speed up the application start time.

Procedure

1. Open the administrative console.
2. Select **Servers**.
3. Select **Application Servers**.
4. Select the server you want to configure.
5. In the Server Infrastructure area, select **Java and Process Management**.
6. In the Server Infrastructure area, select **Process Definition**.
7. In the Additional Properties area, select **Java Virtual Machine**.
8. In the Additional Properties area, select **Custom Properties**.
9. Select the **New** box.
10. In the **Name** entry field, type *com.ibm.websphere.ejbcontainer.initializeEJBsAtStartup*.
11. In the **Value** entry field, type *true*. Entering *true* causes all Enterprise JavaBeans to initialize when your application starts. Entering *false* causes initialization of all beans to be delayed.

    Note: Setting com.ibm.websphere.ejbcontainer.initializeEJBsAtStartup to either true or false takes precedence over any *Start EJB at Application Start* settings made on individual EJB types.
12. Select **OK**.

What to do next

This task can also be done by using an assembly tool.

Changing applications to WebSphere "version specific" setRollbackOnly behavior

Use this task to allow post-EJB 3.0 applications to exhibit the pre-EJB 3.0 behavior and to allow pre-EJB 3.0 applications to exhibit post-EJB 3.0 behavior. The steps in this task that provide this processing behavior are based on a very specific processing scenario, which is explained below.

About this task

Processing Scenario: The basis for this task is the following processing scenario.

An EJB method is invoked which starts a global transaction. Within the execution of this method another EJB method is invoked that continues to run within the same transaction. During the execution of this method, the setRollbackOnly() method is invoked. The EJB Container behavior when this scenario is encountered is dictated by the EJB Specification. However, this behavior was required to change in the WebSphere Application Server for EJB 3.0 support.

Section 13.6.2.8 of JSR 220: Enterprise JavaBeans™,Version 3.0 EJB Core Contracts and Requirements document, as well as previous versions of the EJB specification, indicates that

"If the container initiated the transaction immediately before dispatching the business method to the instance (as opposed to the transaction being inherited from the caller), the container must note that the instance has invoked the setRollbackOnly method. When the business method invocation completes, the container must roll back rather than commit the transaction. If the business method has returned normally or with an application exception, the container must pass the method result or the application exception to the client after the container performed the rollback." also section 14.3.11 states: "However, the container should not throw the javax.ejb.EJBException or java.rmi.RemoteException or if the container performs a transaction rollback because the instance has invoked the setRollbackOnly method on its EJBContext object. In this case, the container must rollback the transaction and pass the business method result or the application exception thrown by the business method to the client."

Historically, the WebSphere Application Server has interpreted the above sections of the specification to dictate that the setRollbackOnly behavior should only applied if the transaction was marked as RollbackOnly within the method that began the transaction. However, the Compatibility Test Suite for the EJB 3.0 specification requires that a transaction marked as RollbackOnly exhibit the above setRollbackOnly behavior, regardless of whether the transaction was marked as RollbackOnly within the method that began the transaction, or within another method within the same transaction that was invoked from the original EJB method.

To illustrate this requirement, consider the following example:

- An application invokes a method, `art`, on EJB A with Container Managed Transaction support, TX_REQUIRED.
- The container begins a transaction and invokes the method.
- Method `A.art()` invokes a method, `bob`, on EJB B with Container Managed Transaction support, TX_REQUIRED.
- Within `B.bob()` the setRollbackOnly method is invoked on the transaction and then completes.

Behavior prior to EJB 3.0: Since the transaction is not initiated with the `B.bob()` method, a TranactionRolledbackException is thrown to `A.art()` and eventually to the client application.

Behavior introduced in EJB 3.0 and beyond: Method `B.bob()` returns normally to method `A.art()`. Method `A.art()` performs a rollback on the transaction and return the results to the client with no exception thrown to the client application, as indicated by the EJB specification.

Since this processing introduces a change in behavior for applications that are migrated to EJB 3.0 or beyond from a previous version, the following JVM system properties are available to change this behavior to suit your requirements:

- `com.ibm.websphere.ejbcontainer.limitSetRollbackOnlyBehaviorToInstanceFor` : This property allows the user to specify application names in which they want to have the EJBs in their EJB 3.0 modules demonstrate the pre-EJB 3.0 setRollbackOnly behavior described above.
- `com.ibm.websphere.ejbcontainer.extendSetRollbackOnlyBehaviorBeyondInstanceFor` : This property allows the user to specify application names in which they want to have the EJBs in their pre-EJB 3.0 modules demonstrate the EJB 3.0 setRollbackOnly behavior described above.

The values of these properties are set to the application names (*appName1:appName2:appName3*) that need to demonstrate the desired behavior.

Procedure

1. Open the administrative console.
2. Select **Servers**.
3. Select **Server Types**.
4. Select **WebSphere application servers**.
5. Select the server that you want to configure.
6. Under Server infrastructure, select **Java and Process Management > Process Definition**.
7. Under Additional properties, select **Java virtual machine > Custom properties > New**.
8. In the **Name** entry field, type the JVM system property.
   com.ibm.websphere.ejbcontainer.limitSetRollbackOnlyBehviorToInstanceFor or
   com.ibm.websphere.ejbcontainer.extendSetRollbackOnlyBehaviorBeyondInstanceFor
9. In the **Value** entry field, type **(appName1:appName2:appName3....).** . Names of the application(s) that the behavior should apply.
10. Select **OK**.
11. Save the configuration.

12. Restart the server.

EJB cache settings

Use this page to configure and manage the cache for a specific Enterprise JavaBeans (EJB) container. To avoid errors from attempting to overload the cache, determine the cache absolute limit. Multiply the number of enterprise beans active in any given transaction by the total number of concurrent transactions expected. Then, add the number of active session bean instances. This value is the limit that the cache will hold. You can use the Tivoli Performance Viewer to view bean performance information.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > **server** > **EJB Container Settings** > **EJB cache settings**.

Cleanup interval

Specifies the interval at which the container attempts to remove unused items from the cache in order to reduce the total number of items to the value of the cache size. This setting applies to the cache only.

The cache manager tries to maintain some unallocated entries that can be allocated quickly as needed. A background thread attempts to free some entries while maintaining some unallocated entries. If the thread runs while the application server is idle, when the application server needs to allocate new cache entries, it does not pay the performance cost of removing entries from the cache. In general, increase this parameter as the cache size increases.Timeouts are specified according to the transaction type:

- Container-managed transactions: The bean provider configures the timeout attribute in the deployment desecriptor.
- Bean-managed transaction: An application calls the UserTransaction.setTransactionTimeout method in the codes.

Data type	Integer
Units	Milliseconds
Range	0 to 2 147 483 647
Default	3000

Cache size

Specifies the number of buckets in the active instance list within the EJB container.

A bucket can contain more than one active enterprise bean instance, but performance is maximized if each bucket in the table has a minimum number of instances assigned to it. When the number of active instances within the container exceeds the number of buckets, that is, the cache size, the container periodically attempts to reduce the number of active instances in the table by passivating some of the active instances. For the best balance of performance and memory, set this value to the maximum number of active instances expected during a typical workload.

Data type	Integer
Units	Buckets in the hash table
Range	Greater than 0. The container selects the next largest prime number equal to or greater than the specified value.
Default	2053

Container interoperability

Container interoperability describes the ability of the product clients and servers at different versions to successfully negotiate differences in native Enterprise JavaBeans (EJB) finder methods support and Java EE compliance.

Interoperability of the handle formats in WebSphere Application Server, Version 5 and Version 5.0.1

Applications that attempt to persist handles to enterprise beans and **EJBHome** needed to subclass ObjectInputStream in WebSphere Application Server, Version 5. This action was required so that the subclass ObjectInputStream could utilize the context class loader to resolve the classes for enterprise beans and EJBHome stubs.

In addition, handles created and persisted in WebSphere Application Server, Version 5 only work with objects that have an unchanged remote interface. If the remote interface is changed, the handle is no longer valid because the stub is serialized inside the handle and its serial Version UID changes if the remote interface changes.

This release introduces a new handle persistence mechanism that avoids the implementation drawbacks of the previous version. However, if handles are used for this WebSphere Application Server deployment, you should consider the following issues when applying this update, future WebSphere Application Server Fix Packs and EJB Container cumulative fixes for WebSphere Application Server, Version 5.

If a WebSphere Application Server, Version 5 persisted handle or home handle is encountered by a WebSphere Application Server, Version 5.0.1 system, it can be read and utilized. In addition, it will be converted to WebSphere Application Server, Version 5.0.1 format if it is re-persisted. The WebSphere Application Server, Version 5.0.1 format cannot be read by a WebSphere Application Server, Version 5 system unless PQ72184 is applied.

Problems arise when handles are persisted and shared across systems that are not at the WebSphere Application Server, Version 5.0.1 level or later. However, a Version 5 system can receive a handle from Version 5.0.1 remotely through a call to get a handle on an enterprise bean or a getHomeHandle on an **EJBHome**. The remote call will succeed, however, any attempt to persist it on the Version 5 system will have the same limitations regarding the use of ObjectInputStream and changes in remote interface invalidating the persisted handle.

When your application stores handles persistently and shares this persistence with multiple clients or application servers, apply WebSphere Application Server, Version 5.0.1 or PQ72184 to both the client and server systems at the same time. Failure to do so can result in the inability of these systems to read the handle data stored by upgraded systems. Also, handles stored by the WebSphere Application Server, Version 5 can force the applications of the updated system to still subclass `ObjectInputStream`. Applications using the WebSphere Application Server Enterprise, Version 5 scheduler and process choreographer, are affected by these changes. These users should update their Version 5 systems at the same time with either Version 5.0.1 or PQ72184.

If the applications store handles in the session context, or locally in a file on the same system, that is not shared by other applications, on different systems, they might be able to update their systems individually, rather than all at once. If Client Container and thin client applications do not share persisted handle data, they can be updated as needed as well. However, handles created and persisted in WebSphere Application Server, Version 5, Version 4.0.3 and later (with the property flag set), or Version 3.5.7 and later (with the property flag set) are not usable if either the home or the remote interface changes.

If any WebSphere Application Server, Version 3.5.7 or Version 4.0.3 and later enables the system property com.ibm.websphere.container.portable to **true**, any handles to objects on that server have the same interoperability limitations. In addition, if any WebSphere Application Server, Version 3.5.7 and later or Version 4.0.3 applications store a handle obtained from a WebSphere Application Server, Version 5 or Version 5.0.1, the same restrictions apply, regarding the need to subclass ObjectInputStream and the usability of handles after a change to the remote interface is made.

Replication of the Http Session and Handles

This note applies to you if you place Handles to Homes or Enterprise JavaBeans, or EJB or EJBHome references in the Http Session in your application and you use Http Session Replication. If you intend to replicate a mixed environment of Version 5.0.0 and Version 5.0.1 or 5.0.2 machines you should first apply the latest Version 5.0.0 container cumulative e-fix to the Version 5.0.0 machines before allowing the Version 5.0.1 or 5.0.2 server into the typology. The reason for this is that Version 5.0.0 servers are not able to understand the persisted Handle format used on the Version 5.0.1 and 5.0.2 server. This is similar to the case of Version 5.0.0 and Version 5.0.1 or 5.0.2 systems trying to use a shared database, mentioned above. But in this case, it is the Http Session object and not the database providing the persistence.

Top Down Deployment Mapping

The size of the Handle objects has grown due to the fix put in to allow serialization and deserialization to occur without the previous requirements of subclassing the ObjectInputStream and so on. Top down deployment of an object that contains EJB and EJBHome references create a database table ddl that has a field of 1000 bytes of VARCHAR for BITDATA which will contain the Handle. It might be that your object's Handle does not fit in the 1000 byte default field, and you might need to adjust this to a higher value. You might try increments of 250 bytes, that is, 1250, 1500, and so on.

Configuring a timer service

You can configure and manage the EJB timer service for a specific EJB container.

About this task

WebSphere Application Server implements the Enterprise JavaBeans (EJB) Timer Service. Based on your business needs, you can use persistent timers or non-persistent timers. Persistent timers are helpful if you are creating a timer for a time-based event that requires assurance of timer existence beyond the life cycle of the server to persist through server shutdowns and restarts. Previously started persistent timers automatically start when your server starts, and they require a database instance.

Non-persistent timers do not use a data store and are canceled when the application server is stopped or fails to remain in an active state. Non-persistent timers exist only on the server where the timer is created.

Both persistent and non-persistent timers require a work manager. Persistent timers use the work manager that is used by the scheduler service. Non-persistent timers use the server default work manager by default, but can be configured to use another work manager using EJB timer scripting. Non-persistent timers do not use the scheduler service.

You can configure and manage the EJB timer service for persistent and non-persistent timers in the administrative console. The configuration for persistent and non-persistent timers is not mutually exclusive. Your application might contain both persistent and non-persistent timers.

Procedure

1. Click **Servers** > **Application servers** > **server_name** > **EJB Container settings** > **EJB timer service settings**.
2. Configure the persistent EJB timer support.
3. If you want to use the internal, or pre-configured, scheduler instance, select **Use internal EJB timer service scheduler instance**. If you choose not to change the default settings, this instance for the scheduler is associated with an Apache Derby database. If you choose to customize the pre-configured instance, complete the following actions:
   a. To change the data source, enter your **Data source JNDI name**. You can use any supported database, such as DB2 or Oracle.
   b. Enter your chosen **Data source alias**.

c. Enter your chosen **Table prefix** if you want to have several server processes use the same database, but different tables.
d. Enter a **Poll interval** value in milliseconds.
e. If you want more timers to run concurrently, enter a new value for **Number of timer threads**.

For more details, see information about timer service settings.

4. If you want to configure your own scheduler instance instead of using the pre-configured internal one, select **Use custom scheduler instance**. You might want to use your own instance to:
   - Change scheduler service configuration options not available for customization on this panel
   - Keep EJB timer tasks in the same database tables as your other tasks
   - Have a single scheduler instance handle all the EJB timers in a cluster. This way, an *ejbTimer* task created on one cluster member can run on a different cluster member.

   To use your own instance, you must:
   a. Configure a scheduler instance through the scheduler service graphical user interface. See the using schedulers documentation for information about how to do this.
   b. Select your **Scheduler JNDI name** from the list.
5. Configure the non-persistent timer support. Support for non-persistent timers is configured in addition to (not instead of) support for persistent timers.
   a. Enter your chosen **Maximum number of retries**.
   b. Enter your chosen **Time interval between retries**.
   c. Select the **Share thread pool configured for persistent timers** or the **Create a separate thread pool for non-persistent timers** option. If you choose the **Create a separate thread pool for non-persistent timers** option, enter your chosen **Number of timer threads**.
6. Optional: Configure data caching for your EJB timers. Caching allows the application server to reuse timer data without having to query the database each time that data is required. See the topic on caching data for a timer service for information on configuring this feature.
7. Click **Apply**.
8. Click **OK**.

Caching data for a timer service

If you want to optimize database access and SQL calls for an EJB timer service, you can enable the application server to cache data for that timer. Caching allows the application server to reuse timer data without having to query the database each time that data is required.

About this task

By allowing the application server to cache data for an Enterprise JavaBeans (EJB) timer service, you can minimize the number of SQL statements that would be generated for calls to methods on the timer interface. When this feature is enabled, data for the timer will be cached in the timer object when you create that timer object.

The specification for enterprise beans requires that an SQL call to the database be made for each call to the javax.ejb.Timer interface, so that the application can ensure that the EJB timer is using the most current data that is available. If these methods are called often, or you have many EJB timers that call any one of these methods, the application server would be generating many SQL statements in a very short amount of time. In some cases, you might find that strict adherence to this requirement is detrimental to performance and causing more overhead than is warranted.

For example, consider a case in which a timer expires only at 12:00 AM every Monday morning. During the course of the week, any applications that call methods on the timer interface will result in the creation of an SQL call, but the call will always return the same data. In addition, when an application calls the ejbTimeout method for a timer, the data that is associated with that timer cannot change; the timer's data

that is stored in the database cannot be updated while an ejbTimeout method is running for that timer. Therefore, any subsequent method calls that applications make during the timeout period will cause the generation of an unnecessary SQL call and a wasted trip to the database.

If you enable caching for timer data, however, the application server will only query the database the first time a configured method is called. For any subsequent calls to one of the configured methods, the application server will use the cached data for the life of the timer object, and a new SQL call will not need to be generated.

Note: Be aware of the following conditions:

- When this feature is enabled, your configuration will not be compliant with the EJB specification.
- This feature can lead to the potential for a timer to use stale data. Cached data for a timer service will become stale after the next expiration period for the timer has passed.

  For example, assume you have a timer that is configured to expire every hour. If you create and save the timer object, the data that the application server caches for that object will only be current for one hour. The cached data would be the same as the data that is stored in the database.

  If you call any of the timer methods within the first hour, the timer data is current. After that expiration period, however, the cached data for the timer becomes stale and might not reflect the data that is actually in the database. In this example, if you queried the EJB container for all timers after one and one half hours, the application would get back a new timer object that contains cached data that will be current for thirty minutes; This is because at hour two the timer will expire again, and any cached data stale will then be stale.

Procedure

1. In the application server's administrative console, select the server that you want to configure. Click **Servers** > **Application Servers** > ***server***.
2. In the Server Infrastructure area, select **Java and Process Management** > **Process Definition**.
3. In the Additional Properties area, select **Java Virtual Machine**.
4. In the Additional Properties area on the Java Virtual Machine panel, select **Custom Properties**.
5. Create or modify the com.ibm.websphere.ejbcontainer.allowCachedTimerDataFor custom property.
   - If the property does not exist, click **New** to create the property.
   - If the property already exists, select the property and click **Modify** to change the values.
6. Enter `com.ibm.websphere.ejbcontainer.allowCachedTimerDataForname` for the name of the custom property.
7. Determine the best value for your needs and environment. The value for this property allows you as much control over caching as you need. You can specify a wildcard or use specific timer names, and you can fine tune control over which methods are cached by assigning integer values based upon those methods.

   These integer values allow you to have precise control over which methods you allow to use cached data. Integers are assigned on a per bean basis or to all beans if you use an asterisk (*). The integer values and their corresponding methods are:

   1 Specifies the getHandle() method

   2 Specifies the getInfo() method

   4 Specifies the getNextTimeout() method

   8 Specifies the getTimeRemaining() method

   16 Specifies the getSchedule() method

   32 Specifies the isPersistent() method

   64 Specifies the isCalendarTimer() method

Note: You can apply this property to any individual methods listed above by using the assigned integer value, or you can apply it to a combination of these methods. For example:

- To apply this property to the getHandle method and the getTimeRemaining method, sum the integer value for the two methods, and use the value 9 (1+8).
- To apply the property to all seven methods, use the value 127 (1+2+4+8+16+32+64), or you could not specify an integer. The default behavior when this property is set is to apply caching to all methods.

When you make your decision, use a wildcard to specify all timers, or specify the JNDI names of the timer beans for which the application server will apply caching.

- You can enter an asterisk (*), which will configure all timers, optionally followed by an integer value - or summation of integer values - that indicate the applicable method on the timer interface. If you do not specify an integer value, the application server will apply caching to all methods.

  For example, the following value applies caching to methods for all timers:

  ```
  *
  ```

- You can enter the name of the EJB timer bean, optionally followed by an integer value - or summation of integer values - that indicate the applicable method on the timer interface. To enter more than one timer bean, separate each application name with a colon (:). If you do not specify an integer value, the application server will apply caching to all methods.

  For example, the following value applies caching to all methods for MyTimerBean1 and MyTimerBean2:

  ```
  MyApp1#MyEJBModule1#MyTimerBean1:MyApp2#MyEJBModule2#MyTimerBean2
  ```

8. Select **OK**.
9. Restart the application server.

Example

The following examples demonstrate different ways to implement the caching feature for timer services. Assume that you have two applications, and each application has timers. The timers use the following J2EE names:

- App1#EJBJar1.jar#EJBTimer1
- App2#EJBJar2.jar#EJBTimer2

The examples will show the value to use for the com.ibm.websphere.ejbcontainer.allowCachedTimerDataForname custom property.

Example 1

This example applies caching to all methods, for all timers, and for all applications.

Use one of the following values:

- Using the defaults:

  ```
  *
  ```

- Specifying an integer:

  ```
  *=127
  ```

Example 2

This example applies caching to the getInfo method on all timers for all applications.

Use the following value:

```
*=2
```

Example 3

This example applies caching to the getHandle and getNextTimeout methods on EJBTimer2.

Use the following value:

App2#EJBJar2.jar#EJBTimer2=5

Example 4

This example applies caching to:

- the getInfo method on EJBTimer1
- the getNextTimeout and getTimeRemaining methods on EJBTimer2

Use the following value:

App1#EJBJar1.jar#EJBTimer1=2:App2#EJBJar2.jar#EJBTimer2=12

Configuring the timer service using scripting

Use wsadmin scripting to configure the Enterprise JavaBeans (EJB) timer service.

Before you begin

You must have a working knowledge of Jacl or Jython and wsadmin scripting.

About this task

The behavior for EJB timers is configured using the EJBTimer configuration object in the server.xml file. If you have EJB timers, you must update the EJBTimer configuration object to obtain the optimal settings for your environment.

The EJBTimer configuration object exists at the server level. This means that each server in a multi-server environment has its own EJBTimer configuration object and must be configured individually.

Procedure

1. Launch the scripting tool using the Jython or scripting language.
2. Determine the attributes on the EJBTimer configuration object that must be updated. You can update the following attributes on the EJBTimer configuration object:
   - datasourceJNDIName
   - datasourceAlias
   - tablePrefix
   - pollInterval
   - numAlarmThreads
   - schedulerJNDIName
   - numNPTimerThreads
   - nonPersistentTimerRetryCount
   - nonPersistentTimerRetryInterval
   - uniqueTimerManagerForNP

   For a complete description of each attribute, see information about EJB timer service settings.

   Four types of EJB timers exist:

   - Persistent timers, supported by a default internal scheduler instance.
   - Persistent timers, supported by a custom scheduler instance.
   - Non-persistent timers, sharing a thread pool with persistent timers.
   - Non-persistent timers, not sharing a thread pool with persistent timers.

   The server is always configured to use one of the two types of persistent timers, and one of the two types of non-persistent timers.

   The EJBTimer configuration object contains the configuration data for all four types of EJB timers. Each of the four types of timer uses a subset of the configuration attributes on the EJBTimer configuration object. All the attributes on the EJBTimer configuration object are used to configure at

least one of the timer types, and none of the attributes are used to configure all the timer types. Thus, you must understand which type of timer you are configured to use, and which configuration attributes apply to that type of timer.

Table 42. Timer types and configuration attributes. Indicates the EJBTimer attributes that are used to configure each type of timer.

Attribute	Persistent, default scheduler	Persistent, custom scheduler	Non-persistent, shared thread pool	Non-persistent, unique thread pool
datasourceJNDIName	Yes	No, specified on custom scheduler configuration instead	No	No
datasourceAlias	Yes	No, specified on custom scheduler configuration instead	No	No
tablePrefix	Yes	No, specified on custom scheduler configuration instead	No	No
pollInterval	Yes	No, specified on custom scheduler configuration instead	No	No
numAlarmThreads	Yes	No	Yes	No
schedulerJNDIName	No	Yes	No	No
numNPTimerThreads	No	No	No	Yes
nonPersistentTimerRetryCount	No	No	Yes	Yes
nonPersistentTimerRetryInterval	No	No	Yes	Yes
uniqueTimerManagerForNP	No	No	Yes	Yes

The presence of a value for the schedulerJNDIName attribute determines which type of persistent timer is used. If the schedulerJNDIName attribute has a value, then a custom scheduler instance is used. If the schedulerJNDIName does not have a value, then the default internal scheduler instance is used.

The numAlarmThreads attribute maps to the Number of timer threads option in the Persistent EJB timer configuration section of the administrative console. The numNPTimerThreads attribute maps to the Number of timer threads option in the Non-persistent EJB timer configuration section of the administrative console.

The uniqueTimerManagerForNP attribute maps to the Share thread pool configured for persistent timers and Create a separate thread pool for non-persistent timers options in the administrative console.

The uniqueTimerManagerForNP attribute determines if the thread pool is shared between persistent and non-persistent timers. It also determines if the numAlarmThreads or numNPTimerThreads configuration attribute is used.

Table 43. The uniqueTimerManagerForNP attribute impacts. The uniqueTimerManagerForNP attribute affects both thread pool sharing and thread configuration.

uniqueTimerManagerForNP attribute	Persistent and non-persistent timers share a thread pool	Thread configuration attribute that is used	Thread configuration attribute that is ignored
true	No	numNPTimerThreads	numAlarmThreads
false	Yes	numAlarmThreads	numNPTimerThreads

3. Obtain a reference to the correct EJBTimer configuration object and store it in a variable.

   Using Jacl:

   ```
   set timer [$AdminConfig list EJBTimer]
   ```

   Using Jython:

   ```
   timer = AdminConfig.list('EJBTimer')
   ```

   If you have a multi-server environment, then multiple EJBTimer configuration objects are returned. Programmatically loop over the list and select the EJBTimer configuration object that corresponds to the server you must update.

In a multi-server environment, as an alternative to programmatically looping over the list of EJBTimer objects, you can manually select the correct EJBTimer object and copy and paste it into your variable.

For example, if the output of your `AdminConfig list` command is:

```
(cells/myCell01/nodes/myCellManager01/servers/dmgr|server.xml#EJBTimer_1)(cells/myCell01/
nodes/myNode02/servers/server1|server.xml#EJBTimer_1246050925244)
```

Copy and paste the reference for the needed EJBTimer object into your variable.

Using Jacl:

```
set timer "(cells/myCell01/nodes/myNode02/servers/server1|server.xml#EJBTimer_1246050925244)"
```

Using Jython:

```
timer = "(cells/myCell01/nodes/myNode02/servers/server1|server.xml#EJBTimer_1246050925244)"
```

4. Update attributes on the EJBTimer configuration object.

   Update attributes on the EJBTimer configuration object using the `AdminConfig modify` command. The first argument to the command is the EJBTimer reference that you obtained in the previous step. The second argument to the command is a list of name-value pairs.

   To set a retry count of 10 attempts, and a retry interval of 15 seconds between each attempt:

   Using Jacl:

   ```
   set update "{nonPersistentTimerRetryCount 10} {nonPersistentTimerRetryInterval 15}"
   $AdminConfig modify $timer $update
   ```

   Using Jython:

   ```
   AdminConfig.modify(timer, '[[nonPersistentTimerRetryCount "10"] [nonPersistentTimerRetryInterval "15"]]')
   ```

5. Save the configuration changes.

   Using Jython:

   ```
   AdminConfig.save()
   ```

   Using Jacl:

   ```
   $AdminConfig save
   ```

6. In a network deployment environment only, synchronize the node.

   Using Jacl:

   ```
   set sync1 [$AdminControl completeObjectName type=NodeSync,node=<your node>,*]
   $AdminControl invoke $sync1 sync
   ```

   Using Jython:

   ```
   sync1 = AdminControl.completeObjectName('type=NodeSync,node=<your node>,*')
   AdminControl.invoke(sync1, 'sync')
   ```

   The node synchronization in these examples must be executed while connected to the server.

Results

As a result of your updates, the EJBTimer configuration object now reflects the attribute values you specified. Restart your server so that the changes are updated on the server.

EJB timer service settings

Use this page to configure and manage the Enterprise JavaBeans (EJB) timer service for a specific EJB container.

To view this administrative console page, click **Servers** > **Server types** > **WebSphere application servers** > ***server name*** > **EJB Container Settings** > **EJB timer service settings**.

Both persistent and non-persistent timers can exist simultaneously, and the persistent and non-persistent configurations are not mutually exclusive. Your application might use both persistent and non-persistent timers.

Use persistent timers when the timer must persist through server shutdowns and restarts. Otherwise, use non-persistent timers when a server shutdown must cancel the timer.

When a persistent timer does not fire because the server is unavailable, then the missed attempt is recovered when the server restarts. When a non-persistent timer does not fire because the server is unavailable, the missed attempt is not recovered, because the server shutdown cancels the non-persistent timer.

Persistent EJB timer configuration

Use internal EJB timer service scheduler instance

The product provides an internal scheduler instance for use by the EJB timer service. The internal scheduler instance is pre-configured for basic EJB timer functionality, and provides limited configuration settings for an EJB timer service.

You can specify that you want to use the internal scheduler instance to manage your persistent timer tasks. They are persisted to a Cloudscape database associated with the server process. Selecting this choice precludes the Use Custom Scheduler Instance option.

The internal scheduler instance is the default. Alternatively, a custom scheduler instance might be used.

Use custom scheduler instance

You can perform a more advanced configuration for the EJB timer service by defining a custom scheduler instance.

A custom Scheduler instance provides more configuration options than the internal EJB timer service pre-configured scheduler instance. You might want to define a custom scheduler instance when running in a clustered environment, allowing all cluster members to run with a single scheduler instance. This definition enables persistent EJB Timers created on one cluster member to run on other cluster members. Providing a custom scheduler instance also enables persistent EJB Timers to be maintained in the same database as other scheduled tasks. Selecting this choice precludes the Use Internal EJB Timer Service Scheduler Instance option.

You might want to define a custom scheduler instance to isolate threads used by the scheduler service from those threads used by the EJB timer service. EJB timer service threads from a custom scheduler instance might be shared for use with non-persistent timers, or you might configure a separate thread pool for non-persistent timers. Even with a thread pool dedicated to EJB timers, timer expirations might fall behind if there are not enough available threads. You must evaluate the number of timers and their expiration frequencies to establish the number of threads.

Data source JNDI name

Specifies the Java Naming and Directory Interface (JNDI) name of the data source where persistent EJB Timers are stored for this EJB container. Any data source available in the name space can be used for EJB Timers.

Multiple EJB containers can share a single data source while using different tables by specifying a table prefix.

Data type String
Default *jdbc/DefaultEJBTimerDataSource*

Data source alias

Specifies an authentication alias to a user name and password used to access the data source.

Data type String

Table prefix

A string prepended to the EJB timer service table names (TASK, TREG, LMGRand LMPR). These tables are created during server start if they do not exist. See the scheduler service for information about manually creating these tables. Multiple independent EJB timer services can share the same database if each instance specifies a different prefix string. If the removeAutomaticEJBTimers command is used to remove timers from a specified scheduler, that scheduler must have a unique table prefix. Otherwise, more timers than expected could be removed.

Data type	String
Default	*EJBTIMER_*

Poll interval

Specifies the interval at which the EJB timer service daemon polls the database. Each poll operation can be expensive. If the interval is small and there are multiple scheduled tasks, polling can use a large portion of system resources. New timers set to expire sooner than this interval might not run until the interval ends. If this value is too large, a potentially large number of timer events might be loaded into memory because all the timer events occurring in the next poll interval are loaded each time.

Data type	Integer
Units	seconds
Default	300
Range	3 to 1800

Number of timer threads

The number of threads used to run concurrent EJB Timer tasks. Setting the number of timer threads to zero disables the EJB timer service.

Data type	Integer
Default	1
Range	0 to 500

Scheduler JNDI name

Specifies the JNDI name of a custom Scheduler instance to use for managing and persisting EJB Timers. This field is only used when you select **Use Custom Scheduler Instance**. Internal EJB timer service scheduler instance configuration information is not applied to the specified scheduler instance.

Data type	String

Non-persistent EJB timer configuration

Maximum number of retries

Specifies the maximum number of times that a failing timeout might be retried. If a timeout is successful upon retry, the server stops attempting to run it. If a retry fails, the server continues to attempt retries until the timeout succeeds, or the timeout limit is reached. Once the retry limit is reached, the server does not attempt to execute the timeout, even if the timeout has not succeeded. The default value of -1 indicates unlimited retries. A value of 0 indicates no retries, and is not specification-compliant. A value of 1 or greater indicates that specific number of retries are allowed.

Data type	Integer
Default	-1
Range	-1 or greater

Time interval between retries

Specifies the interval between retry attempts for a failed timeout. The first retry always occurs immediately, regardless of the interval configured here. All additional retries wait for the interval specified here. A value of 0 indicates that all retries are immediate. A value of 1 or higher indicates that retries must wait for that specific number of seconds.

Data type	Integer
Default	300 seconds
Range	0 or greater

Share thread pool configured for persistent timers

Specifies that non-persistent timers share a thread pool with persistent timers. If persistent timers are using the default internal scheduler instance, the shared thread pool is configured using the configuration settings specified in the Persistent EJB timer configuration section. If the persistent timers are using a custom defined scheduler, the thread pool configuration was specified as part of the configuration for that custom scheduler.

Create a separate thread pool for non-persistent timers

Specifies that non-persistent timers do not share a thread pool with persistent timers. Rather, a unique thread pool is created for non-persistent timers only.

Number of timer threads

Specifies the number of threads available in the unique thread pool used for non-persistent timers. This configuration option is only available when non-persistent timers are not sharing a thread pool with persistent timers. This configuration option is different from the Number of timer threads configuration option in the Persistent EJB timers configuration section because that option applies only to persistent timers using the default internal scheduler instance.

Data type	Integer
Default	1
Range	0 to 500

Managing message-driven beans

You can manage the Java EE Connector Architecture (JCA) Version 1.5-compliant message-driven beans that you deploy as message endpoints, and you can manage the message listener resources for non-JCA message-driven beans that you deploy against listener ports.

Before you begin

For WebSphere Application Server Version 7 and later, listener ports are stabilized. For more information, read the article on stabilized features. You should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications. However, you should not begin this migration until you are sure the application does not have to work on application servers earlier than WebSphere Application Server Version 7. For example, if you have an application server cluster with some members at Version 6.1 and some at Version 7, you should not migrate applications on that cluster to use activation specifications until after you migrate all the application servers in the cluster to Version 7.

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

You can manage the following resources for message-driven beans:

- JCA 1.5-compliant message-driven beans that you deploy as message endpoints, and the associated activation specifications.
- The message listener service, listener ports, and listeners for non-JCA message-driven beans that you deploy against listener ports.

Procedure

- Manage JCA 1.5-compliant message-driven beans that are used as message endpoints.

  JCA 1.5-compliant message-driven beans, deployed by using activation specifications, can be used as message endpoints. You can start and stop specific endpoints within your applications to ensure that messages are delivered only to listening message-driven beans that are interacting with healthy resources.
- Manage message listener resources for message-driven beans.

  The message listener service supports message-driven beans that are used with a non-JCA messaging provider. A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. When you deploy a message-driven bean, you associate the bean with a listener port. When a message arrives on the destination, the listener passes the message to a new instance of a message-driven bean for processing. You can manage the resources used by the message listener service, including being able to start and stop specific listener ports manually.

Managing messages with message endpoints

Manage message delivery for message-driven beans (MDB) that are deployed as message endpoints. The message endpoints are managed beans (MBeans) for inbound resource adapters that are compliant with Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) Version 1.5.

About this task

The application server provides message endpoint MBeans to assist you in managing the delivery of a message to your message-driven beans that are acting as listeners on specific endpoints, which are destinations, and in managing the enterprise information system (EIS) resources that are utilized by these message-driven beans. Message-driven beans that are deployed as message endpoints are not the same as message-driven beans that are configured against a listener port. Message-driven beans that are used as message endpoints must be deployed using an ActivationSpecification that is defined within a resource adapter configuration for JCA Version 1.5.

With message endpoint MBeans, you can activate and deactivate specific endpoints within your applications to ensure that messages are delivered only to listening message-driven beans that are interacting with healthy EIS resources. This capability allows you to optimize the performance of your JMS applications in situations where an EIS resource is not behaving as expected. Message delivery to an endpoint typically fails when the message driven bean that is listening invokes an operation against a resource that is not healthy. For example, a messaging provider, which is an inbound resource adapter that is JCA Version 1.5 compliant, might fail to deliver messages to an endpoint when its underlying message-driven bean attempts to commit transactions against a database server that is not responding.

Note: Design your message-driven beans to delegate business processing to other enterprise beans. Do not access the EIS resources directly in the message-driven bean, but do so indirectly through a delegate bean.

Message endpoint MBeans alleviate two problems that are inherent to applications that provide message endpoints that access resources:

- Failed messages require additional processing, such as delivering them to the listening endpoint again or redirecting them to alternate destinations that process failed messages. In addition, a resource adapter might redeliver a message to an endpoint an infinite number of times.
- Message redirection requires the implementation of specialized destinations (queues and listeners) to process failed messages, as well as the logic to detect message failures. Message redirection is potentially error prone and computationally expensive due to its complexity.

The capability to deactivate (pause) and reactivate (resume) a specific message endpoint alleviates these problems by enabling the administrator to deactivate the endpoint from processing messages that are destined to fail. When the message endpoint is deactivated, you can repair the resource that is causing the problems and reactivate the endpoint to resume handling message requests. In the course of troubleshooting, you will not affect the resource adapter or the application that is hosting the endpoint.

If you are connecting to WebSphere MQ, you can also use the WAS_EndpointInitialState custom property in the activation specification to make the message endpoint start out in a deactivated state. When you set this property to Inactive, the message-driven bean connects with the destination, but does not start receiving messages. Use this setting to automatically deactivate a message endpoint when you know that certain tasks must be completed, services must be started, or checks must be carried out, before message handling begins. You activate the message endpoint in the same way as you would reactivate a message endpoint that you paused during its operation.

Procedure

1. Using the administrative console, navigate to the Message Endpoints panel for the application that is hosting the message endpoint.
   a. Select the **Applications** > **Application Types** > **Websphere enterprise applications** > ***application_name***.
   b. Select the **Runtime** panel.
   c. Select **Message Endpoints**. The panel lists the set of message endpoints that are hosted by the application.
2. Optional: Temporarily disable a message endpoint from handling messages and troubleshoot the problem.
   a. Deactivate the message endpoint by selecting the appropriate endpoint and clicking **Pause**.
   b. When the message endpoint is inactive, diagnose and repair the underlying cause of the delivery failures.
   c. Reactivate the message endpoint by selecting the appropriate endpoint and clicking **Resume**.
3. Optional: Activate a message endpoint that started out in a deactivated state. Select the appropriate endpoint and click **Resume**.

Results

The behavior you will observe when you deactivate (pause) a message endpoint using the message endpoint MBean is dependent upon a variety of factors, including the resource adapter that manages the message endpoint, the configuration of the message endpoint and the application server topology. Some specific examples of interest are as follows:

- **MDB listening on a non-durable topic (dependent on configuration):** The behavior that is implied by the deactivation (pause) of a message endpoint is often dependent upon the function that it is fulfilling. For example, if you have configured a message-driven bean to listen on a non-durable topic on the service integration bus, deactivating the message endpoint is analogous to stopping the application and will cause the subscription to be closed. This means that any messages that are published during the time that the message endpoint is paused will not be received by the message-driven bean.

- **Clustered message-driven bean (dependent on topology):** In this scenario a message-driven bean application has been deployed to a cluster of servers. A given message endpoint MBean controls only the behavior of the MDB in one server from the cluster, so will cause only one server to stop processing messages. Depending upon the messaging configuration and the specific resource adapter in use the messages that would have been consumed by the paused message endpoint may be consumed by the active message endpoints in the cluster, or they may remain unconsumed until the paused message endpoint is resumed.
- **Clustered message-driven bean, a non-clustered queue:** In this scenario, you have a cluster of servers with the same message-driven bean deployed to them. This is similar to the case, in which you have different message-driven beans with the same message selection criteria, except that in this case the message-driven beans are logically the same message-driven bean. Pausing the endpoint will cause only one of the servers to stop receiving messages, and the other message-driven beans will receive all the messages; none of the messages will be orphaned. To stop all of the endpoints, you must direct each server in the cluster to stop the local message endpoint.
- **Clustered message-driven bean, clustered queue:** In this scenario, each message-driven bean is pulling messages from a different partition of the queue. Messaging through WebSphere MQ and the Service Integration Bus have similar, but different, capabilities. If you are using WebSphere MQ, then pausing one endpoint will not allow the other instances of the message-driven bean to receive the messages. In the Service Integration Bus, messages from a paused endpoint will be redirected to the other message-driven beans.

Managing message listener resources for message-driven beans

Manage the resources used by the message listener service to support message-driven beans, typically for use with a messaging provider that does not have a Java EE Connector Architecture (JCA) 1.5 resource adapter.

Before you begin

For WebSphere Application Server Version 7 and later, listener ports are stabilized. For more information, read the article on stabilized features. You should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications. However, you should not begin this migration until you are sure the application does not have to work on application servers earlier than WebSphere Application Server Version 7. For example, if you have an application server cluster with some members at Version 6.1 and some at Version 7, you should not migrate applications on that cluster to use activation specifications until after you migrate all the application servers in the cluster to Version 7.

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

The message listener service is an extension to the JMS functions of the JMS provider and provides a listener manager, which controls and monitors one or more JMS listeners. Each listener monitors either a JMS queue destination (for point-to-point messaging) or a JMS topic destination (for publish/subscribe

messaging). A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. When you deploy a message-driven bean, you associate the bean with a listener port. When a message arrives on the destination, the listener passes the message to a new instance of a message-driven bean for processing. For more information, see Message-driven beans - listener port components.

Procedure

1. Configure the message listener service.
2. Administer listener ports.

   You can complete any of the following administrative tasks:
   - Create or configure a listener port.
   - Start or stop a listener port.
   - Delete a listener port.
3. Configure security for message-driven beans that use listener ports.

Results

You have configured the resources needed by the message listener service to support message-driven beans.

Configuring the message listener service

To support message-driven beans deployed against listener ports, you configure the properties of the message listener service for an application server.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

The message listener service is an extension to the JMS functions of the JMS provider and provides a listener manager, which controls and monitors one or more JMS listeners. Each listener monitors either a JMS queue destination (for point-to-point messaging) or a JMS topic destination (for publish/subscribe messaging). For more information, see Message-driven beans - listener port components.

When you deploy an enterprise application to use message-driven beans with listener ports, you can browse or change the configuration of the message listener service for a given application server.

| If your messaging system is running in non-ASF mode, to avoid unwanted transaction timeouts, you must
| make sure that the time that you specify for the **NON.ASF.RECEIVE.TIMEOUT** message listener service
| custom property is smaller than the sum of the maximum amount of time that the `onMessage()` method of
| the message-driven bean (MDB) takes to process the message, plus the time you specify for the **Total
| transaction lifetime timeout** transaction service property.

Procedure

1. Display the listener service settings page:
   a. In the navigation pane, select **Servers** > **Server Types** > **WebSphere application servers**.
   b. In the content pane, click the name of the application server.
   c. Under Communications, click **Messaging** > **Message Listener Service**.
2. Optional: Browse or change the value of properties for the message-driven bean thread pool.
   a. Click **Thread Pool**.
   b. Change the following properties, as required:

      Minimum size
      The minimum number of threads to allow in the pool.

      Maximum size
      The maximum number of threads to allow in the pool.

      Thread inactivity timeout
      The number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.

      Note: The administrative console does not allow you to set the inactivity timeout to a negative number. To do this you must modify the value directly in the config.xml file.

      Allow thread allocation beyond maximum thread size
      Select this check box to enable the number of threads to increase beyond the maximum size configured for the thread pool.
   c. Click **OK**.
3. Optional: Specify any message listener service custom properties that you need, as **Custom properties** of the message listener service.
   a. Click **Custom properties**
   b. For each custom property, specify the name and value you require.
      If you have not specified a property before:
      1) Click **New**.
      2) Type the name of the property.
      3) Type the value of the property.
      4) Click **OK**.

   For more information about these custom properties, see “Message listener service custom properties” on page 419.
4. Save your changes to the master configuration.
5. To activate the changed configuration, stop then restart the application server.

Results

You have configured the properties of the message listener service for a given application server.

Avoiding transaction timeouts in non-ASF mode:

If your messaging system runs in non-Application Server Facilities (non-ASF) mode, you must configure the **Total transaction lifetime timeout** transaction service property and the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property correctly, to avoid unwanted transaction timeouts.

Before you begin

To carry out the steps in this task, your messaging system must be running in non-ASF mode. To change from ASF mode to non-ASF mode, add the **NON.ASF.RECEIVE.TIMEOUT** custom property to the message listener service as described in “Configuring the message listener service” on page 415.

About this task

For WebSphere Application Server Version 7 and later, listener ports are stabilized. For more information, read the article on stabilized features. You should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications. However, you should not begin this migration until you are sure the application does not have to work on application servers earlier than WebSphere Application Server Version 7. For example, if you have an application server cluster with some members at Version 6.1 and some at Version 7, you should not migrate applications on that cluster to use activation specifications until after you migrate all the application servers in the cluster to Version 7.

If your messaging system is running in non-ASF mode, to avoid unwanted transaction timeouts, you must make sure that the time that you specify for the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property is smaller than the sum of the maximum amount of time that the `onMessage()` method of the message-driven bean (MDB) takes to process the message, plus the time you specify for the **Total transaction lifetime timeout** transaction service property.

Procedure

1. To configure the **Total transaction lifetime timeout** transaction service property, complete step 8 in “Configuring transaction properties for an application server” on page 2491.
2. To configure the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service** > **Custom Properties**.
3. Click **NON.ASF.RECEIVE.TIMEOUT**. The General Properties page is displayed.
4. Modify the **Value** field. The value for **NON.ASF.RECEIVE.TIMEOUT** must be specified in milliseconds. Make sure that the value you specify, when converted into seconds (by dividing by 1000) , is less than the value you have specified for **Total transaction lifetime timeout** combined with the maximum number of seconds the onMessage() method of your MDB takes to process a message.
5. Click **OK**.
6. Stop and restart the application server.

Example

If **Total transaction lifetime timeout** and **NON.ASF.RECEIVE.TIMEOUT** are not correctly configured, transactions can time out before they are completed. This is because the thread begins calling the `receive()` method as soon as the transaction is created. In the following example, **NON.ASF.RECEIVE.TIMEOUT** is set to `110000` milliseconds (110 seconds), **Total transaction lifetime timeout** is set to `120` seconds and the onMessage () method of the MDB takes 15 seconds to process a message. The example supposes that a message does not appear at the destination until the `receive()` method has almost timed out:

1. The listener port starts and allocates a thread from the thread pool and creates a transaction on the thread.
2. The thread calls the `receive()` method to listen for messages.
3. After 110 seconds a message appears at the destination.
4. The thread removes the message from the destination and calls the `onMessage()` method of the MDB to begin processing the message.

5. Ten seconds later, the transaction timeout is reached. The application server marks the transaction for rollback.
6. Five seconds later, the `onMessage()` method finishes processing the message and tries to commit the transaction.
7. The total amount of time that has elapsed since the transaction was started is 125 seconds (110 seconds waiting for a message, plus 15 seconds to process the message). As this time is longer than the transaction timeout, the application server prevents the transaction from being committed, and it is rolled back.

Message listener service:

The message listener service is an extension to the JMS functions of the JMS provider. It provides a listener manager that controls and monitors one or more JMS listeners, which each monitor a JMS destination on behalf of a deployed message-driven bean.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service**

Custom Properties

You can use the Custom properties page to define the following properties for use by the message listener service.

- "DYNAMIC.CONFIGURATION.ENABLED" on page 419
- "MAX.RECOVERY.RETRIES" on page 419
- "MQJMS.POOLING.THRESHOLD" on page 419
- "MQJMS.POOLING.TIMEOUT" on page 420
- "NON.ASF.RECEIVE.TIMEOUT" on page 420
- "NON.ASF.BMT.ROLLBACK.ENABLED" on page 421
- "RECOVERY.RETRY.INTERVAL" on page 421
- "SERVER.SESSION.POOL.REAP.TIME" on page 421
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT" on page 422
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*" on page 422

Listener Ports

You can use the Listener Ports page to create and modify listener ports by specifying the following properties:

- Name
- Initial State
- Description
- Connection factory JNDI name
- Destination JNDI name
- Maximum sessions
- Maximum retries
- Maximum messages

Thread pool

You can use the Thread pool page to change the following properties for the message-driven bean thread pool:

- Minimum size

- Maximum size
- Thread inactivity timeout

Message listener service custom properties:

Use this panel to view or change custom properties of the message listener service.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service** > **Custom Properties**.

You can use the Custom properties page to define the following properties for use by the message listener service.

- "DYNAMIC.CONFIGURATION.ENABLED"
- "MAX.RECOVERY.RETRIES"
- "MQJMS.POOLING.THRESHOLD"
- "MQJMS.POOLING.TIMEOUT" on page 420
- "NON.ASF.RECEIVE.TIMEOUT" on page 420
- "NON.ASF.BMT.ROLLBACK.ENABLED" on page 421
- "RECOVERY.RETRY.INTERVAL" on page 421
- "SERVER.SESSION.POOL.REAP.TIME" on page 421
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT" on page 422
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*" on page 422

DYNAMIC.CONFIGURATION.ENABLED:

This property controls whether the application server on which a listener port is created requires to be restarted. Set this property to true to enable dynamic configuration.

Data type	Boolean
Default	False (not selected)

MAX.RECOVERY.RETRIES:

The maximum number of times that a listener port managed by this service tries to recover from a failure before giving up and stopping. When stopped the associated listener port is changed to the stop state. The interval between retry attempts is defined by the **RECOVERY.RETRY.INTERVAL** property.

A failure can be caused by either of the following conditions:

- An unexpected error has occurred when a listener port tries to get a message from the JMS provider.
- The connection between the application server and the JMS provider has been lost, usually due to a network error.

Data type	Integer
Units	Retry attempts
Default	5
Range	0 (no retries) through 2147483647

MQJMS.POOLING.THRESHOLD:

The maximum number of unused connections in the pool.

An MQSimpleConnectionManager allocates connections on a most-recently-used basis, and destroys connections on a least-recently-used basis. By default, a connection is destroyed if there are more than ten unused connections in the pool.

Data type	Integer
Units	Number of connections
Default	10

MQJMS.POOLING.TIMEOUT:

The number of milliseconds after which a connection in the pool is destroyed if it has not been used.

An MQSimpleConnectionManager allocates connections on a most-recently-used basis, and destroys connections on a least-recently-used basis. By default, a connection is destroyed if it has not been used for five minutes.

Data type	Integer
Units	Milliseconds
Default	5 minutes

NON.ASF.RECEIVE.TIMEOUT:

The timeout in milliseconds for synchronous message receives performed by message-driven bean listener sessions in the non-ASF mode of operation.

Note: The message listener service has two modes of operation, Application Server Facilities (ASF) and non-Application Server Facilities (non-ASF):

- ASF mode provides concurrency and transactional support for applications. For publish/subscribe message-drive beans, ASF mode provides better throughput and concurrency, because in the non-ASF mode the listener is single-threaded.
- Non-ASF mode is mainly for use with third-party messaging providers that do not support JMS ASF, which is an optional extension to the JMS specification. Non-ASF mode is also transactional but, because the path length is shorter than for ASF mode, usually provides improved performance.

To enable the non-ASF mode of operation for all message-driven bean listeners on the application server, set this property to a non-zero value.

If your messaging system is running in non-ASF mode, to avoid unwanted transaction timeouts, you must make sure that the time that you specify for the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property is smaller than the sum of the maximum amount of time that the `onMessage()` method of the message-driven bean (MDB) takes to process the message, plus the time you specify for the **Total transaction lifetime timeout** transaction service property.

Data type	Integer
Units	Milliseconds
Default	ASF mode (custom property not created)
Range	0 or greater milliseconds

0 Non-ASF mode is disabled

1 or more
The timeout in milliseconds for non-ASF message-driven bean listener synchronous session receives

Recommended	If a transaction timeout occurs, the message must recycle causing extra work. If you want to use the non-ASF mode, set this property to less than the transaction timeout, but greater than or equal to the maximum duration of your message-driven bean onMessage() method. For example, if your message-driven bean onMessage() method typically takes a maximum of 10 seconds, and the transaction timeout is set to 120 seconds, you might set the **NON.ASF.RECEIVE.TIMEOUT** property to no more than 110000 milliseconds (that is, 110 seconds).

NON.ASF.BMT.ROLLBACK.ENABLED:

When the non-Application Server Facilities (non-ASF) mode of operation is in use (because you have set the **NON.ASF.RECEIVE.TIMEOUT** property to a non-zero value), and a message-driven bean that uses bean-managed transactions generates a runtime exception, the **NON.ASF.BMT.ROLLBACK.ENABLED** property determines whether messages are returned to the destination.

Note: The message listener service has two modes of operation, Application Server Facilities (ASF) and non-Application Server Facilities (non-ASF):

- ASF mode provides concurrency and transactional support for applications. For publish/subscribe message-drive beans, ASF mode provides better throughput and concurrency, because in the non-ASF mode the listener is single-threaded.
- Non-ASF mode is mainly for use with third-party messaging providers that do not support JMS ASF, which is an optional extension to the JMS specification. Non-ASF mode is also transactional but, because the path length is shorter than for ASF mode, usually provides improved performance.

When this property is set to false (default), the message is automatically acknowledged before it is passed to the message-driven bean.

When this property is set to true, the message listener service sends a message acknowledgement to the client after the message is successfully processed by the message-driven bean, and the message listener service requests recovery of any message for which the bean generates an exception.

Data type	Boolean
Default	False

RECOVERY.RETRY.INTERVAL:

The time in seconds between retry attempts by a listener port to recover from a failure. The maximum number of retry attempts is defined by the **MAX.RECOVERY.RETRIES** property.

A failure can be caused by either of the following conditions:

- An unexpected error has occurred when a listener port tries to get a message from the JMS provider.
- The connection between the application server and the JMS provider has been lost, usually due to a network error.

Data type	Integer
Units	Seconds
Default	60
Range	1 through 2147483647

SERVER.SESSION.POOL.REAP.TIME:

The time in seconds between checks on server session pools. To enable server session pool monitoring, set this property to a non-negative value.

The **SERVER.SESSION.POOL.REAP.TIME** custom property is not applicable if your messaging system is running in non-ASF mode.

Data type	Integer
Units	Seconds
Default	-1 (disabled)
Range	-2147483648 through 2147483647

SERVER.SESSION.POOL.UNUSED.TIMEOUT:

The default server session pool timeout in seconds.

When this property is set to a non-negative value, it is compared to the time that has elapsed since a server session was used. If the timeout value is less than the elapsed time, the server session is removed from the server session pool and its JMS session is returned to the JMS session pool. For example, if the timeout value is one second and the time that has elapsed since a particular server session was used is two seconds, that server session is removed from the server session pool and its JMS session is returned to the JMS session pool.

The **SERVER.SESSION.POOL.UNUSED.TIMEOUT** custom property is not applicable if your messaging system is running in non-ASF mode.

Data type	Integer
Units	Seconds
Default	-1 (disabled)
Range	-2147483648 through 2147483647

SERVER.SESSION.POOL.UNUSED.TIMEOUT.lpname:

This property overrides the default **SERVER.SESSION.POOL.UNUSED.TIMEOUT** value for the listener port with the name defined for *lpname*. This value applies to all message-driven beans that use the specified listener port.

If this override is set to a non-negative value, it overrides the **SERVER.SESSION.POOL.UNUSED.TIMEOUT** property, even if the **SERVER.SESSION.POOL.UNUSED.TIMEOUT** property has a negative value.

If this override is set to a negative value, it disables server session pool monitoring for the specified listener port.

The **SERVER.SESSION.POOL.UNUSED.TIMEOUT.lpname** custom property is not applicable if your messaging system is running in non-ASF mode.

Data type	Integer
Units	Seconds
Default	Not set
Range	-2147483648 through 2147483647

Administering listener ports

You can use the WebSphere Application Server administrative console to administer listener ports, which each define the association between a connection factory, a destination, and a message-driven bean.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. If you set the initial state of a listener port to Started, the listener port is started automatically when a message-driven bean associated with that port is installed.

Listener ports can be manually started and stopped. If a message-driven bean fails to process a message several times, the listener port is automatically stopped by the application server. When a listener port is stopped, the listener manager stops the listeners for all message-driven beans associated with the port. Consequently, the associated message-driven beans can no longer process messages.

Note: You do not usually need to start or stop a listener port manually.

Procedure

- Create a new listener port.

  Create a new listener port, to specify a new association between a connection factory, a destination, and a message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination.
- Configure a listener port.

  Browse or change the configuration properties of a listener port.
- Start a listener port.
- Stop a listener port.
- Delete a listener port.

Creating a new listener port:

You create a new listener port for the message listener service to define the association between a connection factory, a destination, and a deployed message-driven bean. This association enables the message-driven bean to retrieve messages from the associated destination.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination. For more information, see Message-driven beans - listener port components.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**. The “Message listener port collection” on page 425 panel is displayed.
3. Click **New**.
4. Specify the following required properties:

   Name The name by which the listener port is known for administrative purposes.

   Connection factory JNDI name
   The JNDI name for the JMS connection factory to be used by the listener port; for example, `jms/connFactory1`.

   Destination JNDI name
   The JNDI name for the destination to be used by the listener port; for example, `jms/destn1`.
5. Optional: Change other properties for the listener port, as required.
6. Click **OK**.
7. Save your changes to the master configuration.
8. To have the changed configuration take effect, stop then restart the application server.

Results

If you set the initial state of a listener port to Started, the listener port is started automatically when a message-driven bean associated with that port is installed.

Configuring a listener port:

Use this task to browse or change the properties of an existing listener port, which is used by message-driven beans associated with the port to retrieve messages.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications. For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination. For more information, see Message-driven beans - listener port components.

When you deploy an enterprise application to use message-driven beans with listener ports, you can browse or change the configuration of a listener port.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” panel is displayed.
3. Select the name of the listener port that you want to work with. This displays the properties of the listener port in the content pane.
4. Optional: Change properties for the listener port, according to your needs.
5. Click **OK**.
6. Save your changes to the master configuration.
7. To have a changed configuration take effect, stop then restart the application server.

Message listener port collection:

The message listener ports configured in the administrative domain

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination.

This panel displays a list of the message listener ports configured in the administrative domain. You can use this panel to add new listener ports or to change the properties of existing listener ports.

To view this administrative console panel, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Messaging] Message Listener Service** > **Listener Ports**

To manage a listener port, select the check box beside the listener port name in the list and click a button:

Button	Resulting action
Convert to activation specification	Opens a wizard that helps you convert the selected listener port to an activation specification.
New	Accesses the panel to configure a new listener port.
Delete	Deletes the selected listener port or ports.
Start	Starts the selected listener port or ports.
Stop	Stops the selected listener port or ports.

Listener port settings:

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination.

Use this panel to view or change the configuration properties of the selected listener port.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service** > **Listener Ports** > ***listener_port***.

Name:

The name by which the listener port is known for administrative purposes.

Data type String
Default Null

Initial state:

The state that you want the listener port to have when the application server is next restarted

Data type Enum
Units Not applicable
Default Started
Range

Started
When the application server is next started, the listener port is started automatically.

Stopped
When the application server is next started, the listener port is not started automatically. If message-driven beans are to use this listener port on the application server, the system administrator must start the port manually or select the Started value of this property then restart the application server.

Description:

A description of the listener port, for administrative purposes within IBM WebSphere Application Server.

Data type String
Default Null

Connection factory JNDI name:

The JNDI name for the JMS connection factory to be used by the listener port; for example, jms/connFactory1.

Data type String
Default Null

Destination JNDI name:

The JNDI name for the destination to be used by the listener port; for example, jms/destn1.

You cannot use a temporary destination for late responses.

Data type String
Default Null

Maximum sessions:

The maximum number of concurrent sessions that a listener can have with the JMS server to process messages.

Each session corresponds to a separate listener thread and therefore controls the number of concurrently processed messages. Adjust this parameter when the server does not fully use the available capacity of the machine.

Data type	Integer
Units	Sessions
Default	1
Range	1 through 2147483647
Recommended	• For message concurrency, that is to process multiple messages simultaneously, set this property to a value greater than 1. Keep this value as low as possible to prevent overloading client applications. A good starting point for a 100% JMS workload with short transaction times is 2 to 4 sessions per processor. If longer running transactions exist, you might need more sessions, which should be determined by experimentation. The total number of sessions specified in the Maximum Sessions property of all configured listener ports must be less than or equal to the number of threads specified for the Maximum Size property of the message listener service thread pool.

Maximum retries:

The maximum number of times that the listener tries to deliver a message to a message-driven bean instance before the listener is stopped, in the range 0 through 2147483647.

Note: A WebSphere MQ queue has a similar property called the **BackoutThreshold** property. If your listener port is reading from a WebSphere MQ queue, then the retry limit and the behavior when the limit is reached is determined by whichever of these two properties is set to the lower limit:

- If you exceed the WebSphere MQ queue **BackoutThreshold** limit, the message that cannot be delivered is moved to somewhere else by WebSphere MQ (for example, to the WebSphere MQ backout requeue queue or the WebSphere MQ dead letter queue) and the listener port services the next message on the queue. In this case, WebSphere Application Server might not know that the message has not been delivered successfully.
- If you exceed the listener port **maximum retries** limit, the listener port stops. You then manually intervene to investigate the problem, possibly to remove the message from the WebSphere MQ queue then restart the listener port.

Data type	Integer
Units	Retry attempts
Default	0 (no retries)
Range	0 (no retries) through 2147483647

Maximum messages:

The maximum number of messages that the listener can process in one transaction.

If the queue is empty, the listener processes each message when it arrives. Each message is processed within a separate transaction.

For the WebSphere V5 default messaging provider or WebSphere MQ as the JMS provider, if messages start accumulating on the queue then the listener can start processing messages in batches. For third-party messaging providers, this property value is passed to the JMS provider but the effect depends on the JMS provider.

Data type	Integer
Units	Number of messages
Default	1
Range	1 through 2147483647
Recommended	For the WebSphere default messaging providers or WebSphere MQ as the JMS provider, to process multiple messages in a single transaction, set this value to more than 1. If messages start accumulating on the queue, a value greater than 1 enables multiple messages to be batch-processed into a single transaction, and eliminates much of the transaction processing costs for JMS messages. **CAUTION:** • If one message in the batch fails processing with an exception, the entire batch of messages is put back on the queue for processing. • Any resource lock held by any of the interactions for the individual messages are held for the duration of the entire batch. • Depending on the amount of processing that messages need, and if XA transactions are being used, setting a value greater than 1 can cause the transaction to time out. If an XA transaction does time out routinely because processing multiple messages exceeds the transaction timeout, reduce this property to 1 (to limit processing to one message per transaction) or increase your transaction timeout.

Starting a listener port:

Use this task to start a listener port on an application server, to enable the listeners for message-driven beans associated with the port to retrieve messages.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. For more information, see Message-driven beans - listener port components.

A listener is active, that is able to receive messages from a destination, if the deployed message-driven bean, listener port, and message listener service are all started. Although you can start these components in any order, they must all be in a started state before the listener can retrieve messages.

If you set the initial state of a listener port to Started, the listener port is started automatically when a message-driven bean associated with that port is installed. You can also start a listener port manually.

When a listener port is started, the listener manager tries to start the listeners for each message-driven bean associated with the port. If a message-driven bean is stopped, the port is started but the listener is not started, and remains stopped. If you start a message-driven bean, the related listener is started.

Procedure

1. Start the administrative console.
2. If you want the listener for a deployed message-driven bean to be able to receive messages at the port, check that the message-driven bean has been started.
3. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.
4. Select the check box for the listener port that you want to start.
5. Click **Start**.
6. Save your changes to the master configuration.

Stopping a listener port:

Use this task to stop a listener port on an application server, to prevent the listeners for message-driven beans associated with the port from retrieving messages.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. For more information, see Message-driven beans - listener port components.

If a message-driven bean fails to process a message several times, the listener port is automatically stopped by the application server. You can also stop a listener port manually. When a listener port is stopped, the listener manager stops the listeners for all message-driven beans associated with the port. Consequently, the associated message-driven beans can no longer process messages.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.
3. Select the check box for the listener port that you want to stop.
4. Click **Stop**.
5. Save your changes to the master configuration.
6. To have the changed configuration take effect, stop then restart the application server.

Deleting a listener port:

Use this task to delete a listener port from the message listener service, to prevent message-driven beans associated with the port from retrieving messages.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. For more information, see Message-driven beans - listener port components.

To delete a listener port, use the administrative console to complete the following steps:

Procedure

1. Start the administrative console.

2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.
3. Select the check box for the listener port that you want to delete.
4. Click **Delete**.

   This action stops the port (needed to allow the port to be deleted) then deletes the port.
5. Save your changes to the master configuration.
6. To have the changed configuration take effect, stop then restart the application server.

Monitoring server session pools for listener ports:

You can minimize the number of resources that server sessions use by enabling server session pool monitoring and defining the timeout value to be applied to a server session.

About this task

Each listener port uses one or more server sessions, which are held in a server session pool. Each server session is associated with a JMS session, which is taken from the JMS session pool that is associated with the JMS connection factory that the listener port is configured to use.

By default, server session pool monitoring is disabled. When a listener port uses a server session the listener port does not release the server session from the server session pool until the listener port is shut down. This means that the associated JMS session is not released into the JMS session pool until the listener port is shut down, even if the listener port is not processing any messages. Consequently the resources that the JMS session uses, for example TCP/IP connections, can be held for a long time, and this can cause problems for resource-constrained systems.

To minimize the number of resources that server sessions use, you must monitor the server session pools. When you enable server session pool monitoring each server session in each server session pool that a listener port uses is monitored to determine how much time has elapsed since the server session was last used. If the elapsed time is greater than the timeout value that you have configured, the server session is removed from the server session pool and its associated JMS session is returned to the JMS session pool. The returned JMS session can be either reused by another application or closed, depending on your JMS session pool settings. You can also configure additional pooling mechanisms, depending on your JMS provider.

Note: Server session pool monitoring cannot be used if the message listener service is operating in non-Application Server Facilities (non-ASF) mode, that is if the NON.ASF.RECEIVE.TIMEOUT message listener service custom property is set to a non-zero value.

Procedure

To enable server session pool monitoring, configure the following message listener service custom properties on each application server as required.

SERVER.SESSION.POOL.REAP.TIME
: To enable server session pool monitoring, set this property to the time in seconds between checks on server session pools (this must be a non-negative value).

SERVER.SESSION.POOL.UNUSED.TIMEOUT
: To specify the default server session pool timeout, set this property to the required number of seconds for the timeout. When this property is set to a non-negative value, it is compared with the time that has elapsed since a server session was used. If the timeout value is less than the elapsed time, the server session is removed from the server session pool and its JMS session is returned to the JMS session pool. For example, if the timeout value is one second and the time

that has elapsed since a particular server session was used is two seconds, that server session is removed from the server session pool and its JMS session is returned to the JMS session pool.

SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*
To override the default SERVER.SESSION.POOL.UNUSED.TIMEOUT value for the listener port with the name *lpname*, set this property to the appropriate value:

- To override the SERVER.SESSION.POOL.UNUSED.TIMEOUT for the specified listener port, set this property to a non-negative value defining the required number of seconds for the server session timeout for this listener port.
- To disable server session pool monitoring for the specified listener port, set this property to a negative value.

The value that you set for this property applies to all message-driven beans that are using the specified listener port.

Example

For example, consider an application server that is configured with listener ports lp1, and lp2.

The following rules apply:

No properties set
If none of the properties are set, server session pool monitoring is disabled and JMS sessions used by server sessions are not returned to the JMS session pool until the listener port (lp1 or lp2), or its associated message-driven bean, is shut down.

SERVER.SESSION.POOL.REAP.TIME and SERVER.SESSION.POOL.UNUSED.TIMEOUT set
Consider, for example, the following settings:

SERVER.SESSION.POOL.REAP.TIME=60

SERVER.SESSION.POOL.UNUSED.TIMEOUT=120

The server session pool of both listener ports (lp1 and lp2) is checked for inactive server sessions every 60 seconds. If a server session is detected as being inactive for more than 120 seconds, it is removed from the server session pool and its JMS session is returned to the JMS session pool. Taking into account the SERVER.SESSION.POOL.REAP.TIME value, the server session pool could be removed from the session pool between two and three minutes after the server session was last used.

SERVER.SESSION.POOL.REAP.TIME and SERVER.SESSION.POOL.UNUSED.TIMEOUT set, and overrides set for SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*
Consider, for example, the following settings:

SERVER.SESSION.POOL.REAP.TIME=60

SERVER.SESSION.POOL.UNUSED.TIMEOUT=120

SERVER.SESSION.POOL.UNUSED.TIMEOUT.lp2=-1

SERVER.SESSION.POOL.UNUSED.TIMEOUT.lp1=60

The server session pool for listener port lp2 is not checked because it has a negative timeout value. In the server session pool for listener port lp1, any server sessions that are inactive for more than 60 seconds are removed from the server session pool.

Administering applications that use the Java Persistence API

Configure JPA to work in your environment

You have developed your applications to work with Java Persistence API (JPA) and now you must configure your JPA applications to work in your environment.

About this task

You must specify options for your database as a part of configuring JPA applications. The application server manages access to data sources. You can configure the data sources, connection pooling, and Java Transaction API (JTA) service in the administrative console. If you have a specific data source for your application, configure the data source before you install your JPA application.

Procedure

1. Configure your data sources through the administrative console. See the topic, Configuring a JDBC provider and data source.
2. Specify the Java Naming and Directory Interface (JNDI) names for the `<jta-data-source>` and `<non-jta-data-source>` elements. For example to use JNDI lookup:

   ```
   <jta-data-source>jdbc/myJTADataSource</jta-data-source>
   <non-jta-data-source>jdbc/myNonJTADataSource</non-jta-data-source>
   ```

   If you use the component name space method (for example, java:comp/env) for data source retrieval, ensure that your application defines these resource references so that you can use these JNDI names to access the data source. This component name space configuration provides more flexibility if you must alter the configuration for the data source. Otherwise, the standard, direct JNDI is used as the data source name. For more information about using the JNDI interface, see the topic, Developing applications that use JNDI. For example, the `persistence.xml` file would have an entry like the following:

   ```
   <jta-data-source>java:comp/env/jdbc/DataSourceJNDI</jta-data-source>
   ```

   OR

   ```
   <jta-data-source>jdbc/DataSourceJNDI</jta-data-source>
   ```
3. Configure persistence provider support in the application server.
   a. “Configuring the JPA default persistence provider” on page 437.
   b. Optional: “Using third-party persistence providers” on page 440.

What to do next

For more information about the commands, classes or other OpenJPA information, refer to the Apache OpenJPA User Guide.

Configuring a JDBC provider and data source

For access to relational databases, applications use the Java Database Connectivity (JDBC) drivers and data sources that you configure for the application server.

Before you begin

Each vendor database requires different JDBC driver implementation classes for JDBC connectivity. A JDBC provider encapsulates those vendor-specific driver files. Through the data source that you associate with the JDBC provider, an application server obtains and manages the physical connections for transactions between applications and the database.

Attention: If you are accessing a DB2 database, IBM Optim pureQuery Runtime is an alternative to JDBC. For more information on pureQuery, see the topic, Task overview: IBM Optim pureQuery Runtime, in the related links section.

Before starting this task, determine the version of data source that you need according to the API specification of your applications.

- *Data sources (WebSphere Application Server Version 4)* are for use with the Enterprise JavaBeans (EJB) 1.0 specification and the Java Servlet 2.2 specification.

- Data sources of the latest standard version are for use with applications that implement the more advanced releases of these specifications.

Procedure

1. Verify that all of the necessary JDBC driver files are installed on your application server. Consult the article, Data source minimum required settings, by vendor for that information. If you opt to configure a user-defined JDBC provider, check your database documentation for information about the driver files.
2. Create a JDBC provider.

   When you create a JDBC provider from the administrative console, see the topic, Configuring a JDBC provider using the administrative console.

   OR

   Using the wsadmin scripting client, see the topic, Configuring a JDBC provider using the scripting.

   OR

   Using the Java Management Extensions (JMX) API, see the topic, Creating a JDBC provider and data source using the JavaManagement Extensions API.
3. Create a data source.

   From the administrative console, see the topic, Creating a data source using the administrative console.

   OR

   Using the wsadmin scripting client, see the topic, Configuring new data sources using scripting. For V4 data sources, see the topic, Configuring new WAS40 data sources using scripting.

   OR

   Using the JMX API, see the topic, Creating a JDBC provider and data source using the JavaManagement Extensions API.

   Required properties: Different database vendors require different properties for implementations of their JDBC drivers. Set these properties on the WebSphere Application Server data source. Because Application Server contains templates for many vendor JDBC implementations, the administrative console surfaces the required properties and prompts you for them as you create a data source. However, if you script your data access configurations, you must consult the article Data source minimum required settings, by vendor, for the required properties and settings options.
4. Optional: Configure custom properties. Like the required properties, custom properties for specific vendor JDBC drivers must be set on the Application Server data source. Consult your database documentation for information about available custom properties. To configure a custom class to facilitate the handling of database properties that are not recognized natively by the Application Server, refer to the topic, Developing a custom DataStoreHelper class.

   You can also learn about optional data source properties in the Application Programming Guide and Reference for Java for your version of DB2 for z/OS if you use the DB2 Universal JDBC Driver provider.
5. Bind resource references to the data source. See the article, Data source lookups for enterprise beans and web modules.
6. Test the connection (for non-container-managed persistence usage). See the topic, Test connection service.

Results

If you use the DB2 JDBC Universal Driver, you might experience data source failures that the application server JVM log does not document. Check the DB2 database log or the WebSphere Application Server JDBC trace log (if JDBC trace was active). You might find that a bad authentication credential is the cause

of failure. Currently the DB2 JDBC Universal Driver does not identify or surface the errors that are produced by non-valid authentication credentials in a proper or consistent way.

Even if you receive information about a bad credential, check the database and JDBC trace logs. These logs provide more reliable, detailed error data on authentication failures.

Note: The JDBC trace log exists only if the JDBC trace service is active during server start up. Activate the service in the administrative console. For more information, see the topic, Enabling trace at server startup. Specify **WAS.database** as the trace group and select **com.ibm.ws.db2.logwriter** as the trace string.

Configuring data source JDBC providers to use pureQuery in a Java SE environment:

Use this task to configure the application data source Java Database Connectivity (JDBC) provider to use pureQuery to access DB2 in a Java Standard Edition (Java SE) environment.

Before you begin

If you need to use multiple DB2 package collections, see the information center topic, Configure pureQuery to use multiple package collections, before continuing with this task.

About this task

IBM Optim PureQuery Runtime makes use of DB2 packages. These packages include information for one or more Structured Query Language (SQL) statements and are stored in the DB2 catalog. You must first run the wsdbgen command on a Java Persistence API (JPA) application to create the packages. The wsdbgen command creates an XML file containing SQL statement information. This XML file must be included into the application Java archive (JAR) file. The DB2 bind command uses this file as input to create the DB2 package.

Important:

- JPA sets the IBM Optim PureQuery Runtime property `pdq.executionMode` to the value `STATIC`.
- The class path must include the installation location for the IBM Optim PureQuery Runtime. See the information center topic on installing IBM Optim PureQuery Runtime for more information.
- The JPA provider implementation must be JPA for the application server (`com.ibm.websphere.persistence.PersistenceProviderImpl`). The OpenJPA persistence provider does not provide support for pureQuery.
- The wsdbgen command requires the URL of a database. The wsdbgen command forces a synchronize mapping function that creates or alters the required tables. For DB2 zOS, V8 unique indexes and LOB tables must be manually created prior to executing the wsdbgen command.

IBM Optim PureQuery Runtime properties are specified in a `pdq.properties` file in the META-INF directory of the application JAR file. The `pdq.ExecutionMode` property is defaulted to STATIC for JPA applications. You can use the pdqProperties property to use pureQuery in DYNAMIC mode. See the information center topic, Using pureQuery in dynamic mode, for more information. PDQ properties, if specified, pass on to the IBM Optim PureQuery Runtime. See the IBM Optim PureQuery Runtime documentation for list of properties and valid values.

- **wsjpa.jdbc.CollectionId**: String value specifying the collection ID to use. This parameter overrides any collection ID that is used during wsdbgen.

Attention: Read more about the DB2 JAR level compliance for IBM Optim PureQuery Runtime at the IBM Support website: System requirements for IBM Optim PureQuery Runtime for Linux, UNIX, and Windows.

Procedure

1. Update the application data source JDBC provider configuration to include the IBM Optim PureQuery Runtime JAR files. Include the `pdq.jar` and `pdqmgmt.jar` files on the class path in addition to the JDBC driver jar files. Either define a new JDBC provider or modify an existing provider to include the JAR files. The class path must include the installation location for the IBM Optim PureQuery Runtime. See the information center topics on JDBC provider settings and installing IBM Optim PureQuery Runtime for more information.
2. Using the DB2 bind command provided by IBM Optim PureQuery Runtime, bind the XML file to the database. This creates the DB2 packages. Refer to the information center topic on the pureQuery Bind utility for more information.

What to do next

If you want to reconfigure the data source for JDBC, remove the `pdq.jar` and `pdqmgmt.jar` from the class path.

Configuring the default JTA and non-JTA data source JNDI names

The Java Transaction API (JTA) and non-JTA data sources to be used for an application can be specified through the <jta-data-source> and <non-jta-data-source> elements of the `persistence.xml` file within an Enterprise JavaBeans (EJB) module.

About this task

If these elements are not configured in the persistence.xml file, then the default JTA and non-JTA data sources configured for the server are used. These values are null by default. However, the default JTA and non-JTA data source Java Naming and Directory Interface (JNDI) names to be used by applications running on this server can be defined using the administrative console.

Procedure

1. Open the administrative console.
2. Select **Servers** > **Application server** > ***server*** > **Container Services**.
3. Click **Default Java Persistence API settings**.
4. Select the **Default JTA data source JNDI name** or the **Default non-JTA data source JNDI name** field.
5. Click **Apply** to save the configuration.

Associating persistence providers and data sources

Java Persistence API (JPA) applications specify the underlying data source that is used by the persistence provider to access the database.

About this task

The application server provides three methods for defining the data sources in the `persistence.xml` file.

Procedure

- Explicitly specify the Java Naming and Directory Interface (JNDI) name in the `persistence.xml` file, and the application directly references the data source. Switching to another data source requires an update to the `persistence.xml` file.

  JPA has two transactional patterns for accessing a data source:

  – The Java Transaction API (JTA) resource pattern depends on global transactions. The JTA resource pattern is typically used within the scope of an Enterprise JavaBeans (EJB) session facade. This supports the session bean to control transaction and security contexts while JPA handles the persistence mappings. In this case, the application does not use the EntityTransaction interface but relies on the EntityManager enlisted with the global transaction when it is accessed.

– The non-JTA resource pattern is used when dealing with a single resource in the absence of global transactions. The non-JTA resource pattern is typically used within the scope of a web application or an application client. The application controls the transaction with the data source with the EntityTransaction interface.

  Within the application server, the use of the <non-jta-data-source> element requires a special configuration for a non-transactional data source. Data sources that are configured for the application server do not function as a <non-jta-data-source>, because all data sources that are configured by the application server are automatically enlisted with the current transactional context. To prevent this automatic enlistment, add an additional data source custom property `nonTransactionalDataSource=true`:

  1. Select **Resources** > **JDBC** > **Data sources**
  2. Select the name of the data source that you want to configure.
  3. Select **WebSphere Application Server data source properties** from the **Additional Properties** heading.
  4. Select **Non-transactional data source**.
  5. Click **OK**.

Note: The JPA specification assumes that connections are obtained with an isolation level that does not hold long-term locks in the database, such as `READ_COMMITTED`. This might not match the WebSphere Application Server default isolation level, which is `REPEATABLE_READ` for most databases. You can find the level that is used for your database by reading the topic, Requirements for setting isolation level.

If the default for your database is not `READ_COMMITTED`, you can change the default by adding an additional data source custom property webSphereDefaultIsolationLevel.

Table 44. Isolation level values. This table shows valid isolation level values.

Value	Isolation Level
1	READ_UNCOMMITTED
2	READ_COMMITTED (JPA default)
4	REPEATABLE_READ (WebSphere Application Server default)
8	SERIALIZABLE

If the isolation level is set to a value that holds long-term read locks, configure the JPA provider to use Pessimistic Locking instead of the default Optimistic Locking. For the JPA provider included with WebSphere Application Server, you can do this by adding the following properties to `persistence.xml` file:

```
<property name="openjpa.Optimistic"="false"/>
<property name="openjpa.LockManager"=pessimistic"/>
```

The JPA specification mandates that the data sources that are defined in `<jta-data-source>` and `<non-jta-data-source>` elements of a persistence unit register in the JNDI name space. For example, the `persistence.xml` file should contain an entry like the following:

```
<jta-data-source>jdbc/DataSourceJNDI</jta-data-source>
```

- The JPA for WebSphere Application Server solution extends the JNDI data-source implementation to allow you to reference data sources in the component name space. In the EJB or web module deployment descriptor file, this is the <resource-ref> element. You can prefix the data source with java:`comp/env/` so the application indirectly references the data source by using the local JNDI name. In this association, the application does not require updates, you change <resource-ref> to use another data source. See the following example:

  ```
  <jta-data-source>java:comp/env/jdbc/DataSourceJNDI</jta-data-source>
  ```

  Note: The JPA specification does not require implementations to include data sources referenced in local JNDI names, so this method of reference might not be portable to other implementations of JPA.

- You can declare openjpa.Connection* properties in the persistence unit as follows:

```
<property name="openjpa.ConnectionDriverName" value="org.apache.derby.jdbc.EmbeddedDriver" />
<property name="openjpa.ConnectionURL" value="jdbc:derby:target/database/jpa-test-database;create=true"/>
```

OR

You can use alternative standard JPA properties that are equivalent to the OpenJPA properties, such as:

Table 45. Standard JPA 2.0 property equivalents. Standard JPA 2.0 properties and the OpenJPA equivalents.

Standard JPA 2.0	OpenJPA Equivalent
javax.persistence.jdbc.driver	openjpa.ConnectionDriverName
javax.persistence.jdbc.url	openjpa.ConnectionURL

What to do next

For information about configuring data sources, see the topic on creating and configuring a data source.

For information about data sources and JPA, see the section on persistence in the Apache OpenJPA User Guide.

Configuring persistence provider support in the application server

Persistence providers are implementations of the Java Persistence API (JPA) specification and can be deployed in the Java EE compliant application server that supports JPA persistence.

About this task

The Enterprise JavaBeans (EJB) 3.0 and later specifications require that an application server container that supports the EJB 3.0 and later programming model must provide a JPA implementation. This is also referred to as a persistence provider. There are two built-in JPA persistence providers: the JPA persistence provider for the application server and the OpenJPA persistence provider. If an explicit provider element is not specified in the persistence unit definitions, the application server will use the default persistence provider, which is the JPA persistence provider for the application server.

The application server provides both the default persistence provider and the Apache OpenJPA persistence provider to support the open source implementation of JPA and allow for easy migration of existing OpenJPA applications to the application server's solution for JPA.

Procedure

- **Configuring a default persistence provider**

  Use the default persistence provider, or specify a persistence provider for your application. With the application server's JPA persistence provider, you can take advantage of the stability and application extensions that are provided with the application server's implementation of JPA. You can decide which persistence implementation best fits your needs. If an application relies on a specific persistence provider for certain functions and settings, you should specify the provider in the persistence unit definition to avoid any incompatibilities.
- **Using third-party persistence providers**

  Use third-party persistence providers in the application server environment.

Configuring the JPA default persistence provider:

Two persistence providers are included in the product: *JPA for WebSphere Application Server persistence provider* and *Apache OpenJPA persistence provider*. The Java Persistence API (JPA) for WebSphere Application Server persistence provider is the default provider for the product. You can use one of these two providers, or a third-party persistence provider, as the default provider.

About this task

Attention: If you have a default persistence provider, default Java Transaction API (JTA) data source Java Naming and Directory Interface (JNDI) name, and default non-JTA data source JNDI name values that were set in the product before V7.0, through the Java virtual machine (JVM) properties, any change to these values through the administrative console override values that were set with the JVM properties.

These properties include, com.ibm.websphere.jpa.default.provider, com.ibm.websphere.jpa.default.jta.datasource, and com.ibm.websphere.jpa.default.nonjta.datasource. Support for these properties has been deprecated. Any values that were set through these properties are displayed as default values on this panel. These values that are set through the administrative console panel overrides any values set through the JVM properties.

JPA for WebSphere persistence provider

While built from the Apache OpenJPA persistence provider, the JPA for WebSphere Application Server persistence provider contains the following enhancements and differences:

- Static SQL support using the DB2 pureQuery feature
- Access intent support
- Enhanced tracing support
- Version ID generation
- WebSphere product-specific commands and scripts
- Translated message files
- The following table shows how the default values for the JPA for WebSphere Application Server persistence provider configuration properties are different from the Apache OpenJPA provider:

Table 46. Comparison. JPA for WebSphere Application Server persistence provider and Apache OpenJPA provider comparison

Property	Apache OpenJPA default value	JPA for WebSphere Application Server persistence provider default value
openjpa.Compatibility	StrictIdentityValues=false	StrictIdentityValues=true
openjpa.RuntimeUnenhancedClasses	supported	warn
openjpa.DynamicEnhancementAgent	true	false
open.jdbc.DriverDataSource	auto	simple

In addition to the above property overrides, the use of the default JPA for WebSphere Application Server persistence provider also implies the use of the following JPA for WebSphere Application Server classes which override the corresponding classes in Apache OpenJPA:

```
com.ibm.ws.persistence.jdbc.kernel.ConstraintUpdateManager;
com.ibm.ws.persistence.jdbc.kernel.WsJpaJDBCBrokerFactory;
com.ibm.ws.persistence.jdbc.sql.DB2Dictionary;
com.ibm.ws.persistence.jdbc.sql.OracleDictionary;
com.ibm.ws.persistence.jdbc.sql.SQLFactoryImpl;
com.ibm.ws.persistence.jdbc.sql.SQLServerDictionary;
com.ibm.ws.persistence.kernel.WsJpaBrokerImpl;
com.ibm.ws.persistence.kernel.WsJpaFinalizingBrokerImpl;
```

If no JPA provider is configured in the <provider> element of the `persistence.xml` file within an Enterprise JavaBeans (EJB) module, the default JPA provider that is currently configured for this server is used. The product is packaged with the JPA for WebSphere Application Server persistence provider defined as the default provider. However, it is possible to override this default and specify a different default through the administrative console.

You can set your default persistence provider in one of two ways. Use one of the following procedures.

Procedure

1. Select the default JPA provider from a list of providers included with the product.
   a. Open the administrative console.
   b. Click **Servers** > **Application Servers**
   c. Select a *server*.
   d. Click **Container Services** > **Default Java Persistence API Settings**
   e. Select **Select a default persistence provider that is included with WebSphere Application Server**.
   f. Click the slider and select from the list.
   g. Click **Apply** and save the configuration.
2. Specify an alternative default persistence provider through the administrative console
   a. Open the administrative console.
   b. Click **Servers** > **Application Servers**
   c. Select a *server*.
   d. Click **Container Services** > **Default Java Persistence API Settings**
   e. Select **Specify an alternative default persistence provider**.
   f. Enter the fully qualified JPA implementation class name of a JPA persistence provider in the box.
   g. Click **Apply** and save the configuration.

Default Java Persistence API settings:

The Java Persistence API (JPA) specification requires a default provider to be defined. If you have applications that use JPA, it is recommended you use this page to provide default values. To increase the portability of your applications, you can use this page to configure the default JPA settings for applications running on this server instead of defining the <provider> element in each persistence unit in your applications. The JPA settings defined here are used for the persistence unit of an application only when the application does not define the JPA settings for that persistence unit.

Note: Application JPA settings always override the settings on this page.

To view this administrative console page, click **Servers** > **Server types** > **WebSphere application servers** > **server** > **Container Services** > **Default Java Persistence API settings**.

Default persistence provider:

Specify the default persistence provider for the application server container. The default persistence provider can be selected either from a list of providers that are included with the product or a user-specified alternate persistence provider.

Select a persistence provider from the product list or specify a fully package qualified JPA implementation class name of an alternate persistence provider.

Default
com.ibm.websphere.persistence.PersistenceProviderImpl

Note: If an alternate persistence provider is specified as the default, make sure the alternate persistence provider is created in the server. See the information center topic on using a third-party persistence provider.

Default Java Transaction API (JTA) data source Java Naming and Directory Interface (JNDI) name:

Specify the default JTA data source used by persistence units for the application server container.

Select the JNDI name for the data source from the drop down box. The JTA data sources that are currently configured and visible to the application server are available in the drop down box selection.

Default
None

Note: If a default JTA data source is not specified, ensure an appropriate JTA data source is specified in the <jta-data-source> or connection properties field in the <properties> element in the persistence unit.

Default non-JTA data source JNDI name:

Specify the default non-JTA data source used by persistence units for the application server container.

Select the JNDI name for the data source from the drop down box. The data sources that are currently configured, visible to the application server, and are set to "non-transactional" are available in the drop down box selection.

Default
None

Note: Some JPA entity features will require a non-JTA data source to be specified. An example of this is automatic entity identity generation. Ensure a non-JTA data source is configured to match your application needs. For information on configuring a non-JTA data source, see the information center topic on associating persistence units and data sources.

Using third-party persistence providers:

Java Persistence API (JPA) for WebSphere Application Server supports third-party persistence providers in their application server environment.

About this task

Java EE applications that use JPA functions can employ third-party persistence providers other than the providers that are included with the application server. Applications can also specify an Apache OpenJPA provider that is a different version than what is included with the application server, as long as the same version of the JPA specification is supported.

There are two basic means to incorporate third-party providers into an application:
- Embedding the persistence provider inside an application
- Using shared libraries

Depending on your requirements, you can embed a persistence provider inside an application, or place the persistence provider into a shared library.

Procedure
- **Embedding a third-party persistence provider within an application** Sometimes an application design must rely on the implementation of a specific persistence provider. The JPA specification permits embedding of a specific JPA provider within a persistence archive. When building the application, you can assemble a specific version of a provider implementation in the application enterprise archive (EAR) file or in a web application (WAR) file. To embed a third-party persistence provider into an application, you must inspect the application design and all dependent prerequisites. Use the following steps to embed a persistence provider inside an application:
  1. Modify the <provider> element to specify explicitly which persistence provider to use to access the persistence entity.

2. Build the specific version of the third-party persistence provider into the application. To manage the persistence correctly, verify that the EntityManagerFactory and the EntityManager are calling the correct provider. Many providers write startup and version information to standard output, which gets included in the SystemOut.log of the application server. This information can be helpful to determine if a third-party provider is being used by your application.

   Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.
3. To use a third-party JPA provider that was not bundled with the application server, configure the class loader order for your application to load the classes with the application class loader first. This is also required if the third-party JPA provider is defined in a shared library that is assigned to a user-defined class loader on the server. If the class loader is not configured properly, the third-party JPA provider that is included with the application server is not loaded and used by your application.
4. Depending on how you packaged the provider, perform one of the following steps:
   – If you packaged the provider in an EAR file, specify the third-party persistence provider binaries in the Manifest.mf class path in those application modules that needs JPA access.
   – If the provider was bundled in a WAR file, include the necessary provider binaries in the `WEB-INF/lib` directory of the web application.
5. Install your application normally.

This configuration localizes the availability of the provider and limiting it to the application alone. While sometimes necessary, embedding persistence providers increases the application memory footprint accordingly.

- **Using shared libraries to implement a third-party persistence provider** Persistence providers accessed by applications in a global environment can be installed as a shared library for the application server. Depending on your requirements, you can share the library to the server and application. Use the following steps to implement a third-party persistence provider using shared libraries:
  1. Modify the <provider> element to specify explicitly which persistence provider to use to access the persistence entity.
  2. Define the persistence provider in a shared library. See the topic on creating shared libraries for more information.
  3. Associate the shared library with the application class loader, or associate the shared library with the server class loader if the library is accessed by many applications.
  4. Configure the class loader order for your application to load the classes with the application class loader first. This is required if the third-party JPA provider is included in the application or if it is defined in a shared library. If the class loader is not configured properly, the JPA provider that is included with the application server is used by your application instead of the third-party JPA provider.

  When the application starts, the specified persistence provider is resolved. From this point on, all persistence requests are handled by this provider.
- A shared library persistence provider can override an existing persistence provider that is the same type and exists in your application class loader hierarchy. If a shared library persistence provider overrides the persistence provider that is shipped with the product, it is recommended that you use an isolated class loader for the shared library. See the topic, Creating shared libraries, for more information.

Task overview: IBM Optim pureQuery Runtime

IBM Optim pureQuery Runtime provides Java Persistence API (JPA) with an alternative way to access a database. PureQuery supports static Structured Query Language (SQL).

About this task

JPA in the Java EE and Java SE environments provides optional support for the pureQuery runtime environment. PureQuery is a high performance Java data access platform that helps manage applications that access data. PureQuery provides an alternate set of APIs that can be used instead of Java Database Connectivity (JDBC) to access the DB2 and Informix database.

To use this feature on the application server, you must install Data Studio pureQuery runtime version 1.2 or later. If you plan to perform the DB2 bind command from the administrative console, or with the wsadmin tool, you must have pureQuery v1.2 or later. Refer to the IBM Optim pureQuery Runtime information center topic on installing pureQuery Runtime for more information.

You can use pureQuery dynamically. The `pdqxml` file location is specified by the pdqProperties property on the data source or connection URL. For more information, see the topic, Using pureQuery in dynamic mode.

PureQuery uses DB2 packages. These packages consist of information for one or more SQL statements and are stored in the DB2 catalog. To create the packages, the user must first run the wsdbgen command on a JPA application. The wsdbgen command creates a *persistence_unit_name*`.pdqxml` file. This file contains pre-generated SQL statements for Create, Update, Delete, and Retrieve, NamedQueries, and NamedNativeQueries of JPA entities. The *persistence_unit_name*`.pdqxml` file must be bound against database. Associated DB2 packages are generated and the SQL statement is started statically at run time. This *persistence_unit_name*`.pdqxml` file must be included into the application Java archive (JAR) file.

The application server offers support for static SQL for Enterprise JavaBeans (EJB) 2.x and later entity beans with the ejbdeploy SQLj option. With JPA, this feature is offered through pureQuery.

There are several benefits to using pureQuery instead of JDBC and SQLJ. Static SQL offers greater security and control over access to data because applications are only granted authority to execute known SQL. Static SQL offers better resource utilization on the DB2 server because it avoids runtime parsing and optimizing of the SQL statements.

When doing the bind process and when you define your JDBC provider, the following four Java archive (JAR) files must be in the class path:

- `db2jcc_license_cisuz.jar`
- `db2jcc_license_cu.jar`
- `pdq.jar`
- `pdqmgmt.jar`

Attention: Read more about the DB2 JAR level compliance for pureQuery at the IBM Support website: System requirements for IBM Optim pureQuery Runtime for Linux, UNIX, and Windows.

WebSphere Application Server support for pureQuery

Restriction:

- There is no support for the QueryTimeout property specified either through the FetchPlan API or through the property plug-in string for wsjpa.ConnectionFactoryProperties. The QueryTimeout value is ignored if specified.
- There is no support for the QueryTimeout property specified through the FetchPlan API. The QueryTimeout value is ignored if specified.
- OpenJPA large result processing uses JDBC APIs for scrollable cursors.

Important:

- JPA sets the pureQuery property, pdq.executionMode, to the value `STATIC`.

- In addition to the JDBC driver JAR file, the JDBC provider configuration must include the JAR file for the pureQuery runtime environment.
- OpenJPA provides support for application programs to programmatically access and alter the FetchPlan at run time. Altering the fetch plan might result in an SQL that has not been generated by the wsdbgen command at application build time. If this occurs, the SQL is executed dynamically rather than using static SQL from the database package.
- If the user changes the application queries, entity mapping or persistence properties, run the wsdbgen command and bind again. This process generates and binds the updated database packages.
- Input parameter values in JPA queries (with both EJB SQL queries and native SQL queries) cannot be `NULL` values except in the case of update statements `SET` expression values. To search for `NULL` values in a WHERE clause of SELECT, UPDATE or DELETE, then enter the `is null` predicate instead.

Procedure

- Learn how to **"Configuring JDBC providers to use pureQuery to access DB2"**.
- Learn how to **"Configuring JDBC providers to use pureQuery to access Informix" on page 444**.
- Learn how to **"Configuring data source JDBC providers to use pureQuery in a Java SE environment" on page 434**.
- Learn how to **"Using pureQuery in dynamic versus static mode for DB2 and Informix" on page 445**.

Configuring JDBC providers to use pureQuery to access DB2:

Use this task to configure the application data source Java Database Connectivity (JDBC) provider to use pureQuery to access DB2 in a Java EE environment.

Before you begin

If you must use multiple DB2 package collections, see the information center topic configure pureQuery to use multiple package collections before continuing with this task.

About this task

PureQuery uses DB2 packages. These packages consist of information for one or more SQL statements and are stored in the DB2 catalog. You must first run the wsdbgen command on a JPA application to create the packages. The wsdbgen command creates an XML file containing SQL statement information. This XML file must be included into the application Java archive (JAR) file. The DB2 bind command uses this file as input to create the DB2 package .

Important:

- JPA sets the pureQuery property pdq.executionMode to the value `STATIC`.
- The JDBC provider configuration must include the JAR file for the pureQuery runtime environment. This JAR file is in addition to the JDBC driver JAR file. See the information center topic on installing pureQuery run time for more information.
- If this is an XA data source, define a new custom property on the data source where *property_name* = `downgradeHoldCursorsUnderXa` and boolean value = `true`.

Procedure

1. Update the application data source JDBC provider configuration to include the pureQuery runtime JAR file. Either define a new JDBC provider or modify an existing provider to include the following JAR files. See information center topics on JDBC provider settings and installing IBM Optim pureQuery Run time for more information.

- pdq.jar
- pdqmgmt.jar

2. Using the DB2 bind command, bind the XML file to the database. This creates the DB2 packages. There are three ways to do this:
   - Use the wsadmin command. Refer to the information center topic on application management command group for the AdminTask object for more information.
   - Using the administrative console. Refer to the information center topic on SQLj profiles and pureQuery bind files settings for more information.
   - Using the DB2 bind command provided by IBM Optim pureQuery Run time. Refer to the information center topic on the pureQuery Bind utility for more information.

What to do next

If you want to reconfigure the data source for JDBC, remove the pdq.jar and pdqmgmt.jar from the class path.

Configuring JDBC providers to use pureQuery to access Informix:

Use this task to configure the application data source Java Database Connectivity (JDBC) provider to use pureQuery to access Informix in a Java EE environment.

About this task

PureQuery provides heterogeneous batching which performs better than homogeneous batching in JDBC. Applications benefit from heterogeneous batching if updates in a transaction involve many entity types and those entities do not have DB-generated keys. To use pureQuery the JDBC driver must specify the db2jcc.jar file (pureQuery does not work with the Informix legacy JDBC driver, ifxjdbc.jar file) and when using JPA, in the persistence.xml file that is included in the application Java archive (JAR) file must specify the following:

```
<property name="pdqProperties" value="dynamic"/>
```

Important:

- The JDBC provider configuration must include the JAR file for the pureQuery runtime environment. This JAR file is in addition to the JDBC driver JAR file. See the information center topic on installing IBM Optim pureQuery Runtime for more information.
- If this is an XA data source, define a new custom property on the data source where *property_name* = downgradeHoldCursorsUnderXa and boolean value = true.

Procedure

1. Update the application data source JDBC provider configuration to include the pureQuery runtime JAR file. Either define a new JDBC provider or modify an existing provider to include the following JAR files. See information center topics on JDBC provider settings and installing IBM Optim pureQuery Runtime for more information.
   - pdq.jar
   - pdqmgmt.jar
   - db2jcc.jar
2. Ensure that the persistence.xml file included in the application JAR file defines the following property:

   ```
   <property name="pdqProperties" value="dynamic"/>
   ```

   See the topic, Using pureQuery in dynamic mode, for more information.

What to do next

If you want to reconfigure the data source for JDBC, remove the `pdq.jar`, `pdqmgmt.jar`, and `db2jcc.jar` files from the class path and replace these files with the JDBC driver, `ifxjdbc.jar` file.

Using pureQuery in dynamic versus static mode for DB2 and Informix:

Using IBM Optim PureQuery Runtime is another way for Java Persistence API (JPA) to access a DB2 and Informix databases. IBM Optim PureQuery Runtime supports static Structured Query Language (SQL).

Before you begin

Important: BatchLimit is a configurable property. The default value for DB2 is `100` and for Informix the default value is `0`. If you set batchLimit to `0`, batching does not occur. For an application to get heterogenous batching for the Informix database backend, batchLimit is configured in the `persistence.xml` file. An example of how to set batchLimit to `100` is as follows:

```
<property name="openjpa.jdbc.DBDictionary" value="batchLimit=100"/>
```

About this task

The Feature Pack for OSGi Applications and Java Persistence API (JPA) 2.0 introduced support for IBM Optim PureQuery Runtime 2.2.0.2 and later. The new feature added for IBM Optim PureQuery Runtime 2.2.0.3, supports Informix, and DB2 applications to use pureQuery in DYNAMIC mode.

This is achieved by setting up pdqProperties on the data source in the Java Enterprise Edition (Java EE) environment. Or, setting pdqProperties on the connection URL in the Java Standard Edition (Java SE) environment.

If pdqProperties is not defined, pureQuery runs in `compatible` mode, which means that the `pdqxml` file is packaged in the application Java archive (JAR) files.

There are several topics that reference pdqProperties in the IBM Integrated Data Management information center.

Procedure

1. When defining a data source In a Java EE environment, add a custom property pdqProperties and set a string value to valid IBM Optim PureQuery Runtime properties. In the following table, an example of executionMode(DYNAMIC) is shown to use pureQuery dynamic mode. Any valid IBM Optim PureQuery Runtime property can be specified to get dynamic execution. When the pdqProperties contains the pureQuery Xml(*pdqxml-file-location*) property, the IBM Optim PureQuery Runtime uses the SQL in STATIC mode that is found in the pdqxml file. This pureQueryXML property contains the pdqxml file location.

   Attention: It is assumed that the pdqxml file is previously bound against the database.

Table 47. pdqProperties settings for DB2 and Informix examples. pdqProperties settings for DB2 and Informix examples

pdqProperties=	DB2 backend	Informix
executionMode(DYNAMIC)	dynamic execution	dynamic execution

Table 47. pdqProperties settings for DB2 and Informix examples (continued). pdqProperties settings for DB2 and Informix examples

pdqProperties=	DB2 backend	Informix
executionMode(STATIC), pureQueryXML(c:/temp/ItemEJB.pdqxml) **Attention:** executionMODE(DYNAMIC) and executionMode(STATIC) are required for dynamic and static executions. For static execution mode, the pdqxml file specification follows the IBM Optim PureQuery Runtime documentation. pureQueryXML(*pdqxml-file-location*) is one of many ways to specify the pdqxml-file-location.The location of the pdqxml file is the path of the pdqxml file. For example, c:/temp/ItemEJB.pdqxml.	static execution	NA (ignored)

2. In a Java SE environment, pdqProperties can be set on the connectionURL. There are two ways to specify the pdqProperties on the connectionURL:
   a. Use openjpa.connectionProperties For example:

```
<property name="openjpa.ConnectionProperties"
          value="DriverClassName=com.ibm.db2.jcc.DB2Driver,
                                Url='jdbc:db2://localhost:50000/demodb:pdqProperties=
                                    pureQueryXml(C:/wsjpa1/fvt/resources/demo.pdqxml);',
          Username=myid, Password=secret" />
```

   b. Use openjpa.ConnectionURL. For example:

```
<property name="javax.persistence.jdbc.driver" value="com.ibm.db2.jcc.DB2Driver"/>
<property name="javax.persistence.jdbc.url"
          value="jdbc:db2:fvt2:pdqProperties=pureQueryXml(C:/wsjpa1/fvt/resources/demo.pdqxml);"/>
<property name="javax.persistence.jdbc.user" value="myid"/>
<property name="javax.persistence.jdbc.password" value="secret"/>
```

   Attention: The pureQueryXml file location must grant read-write permissions for the IBM Optim PureQuery Runtime to update the file.

   Attention: The data source that is defined to run IBM Optim PureQuery Runtime should not be shared with applications that run SQL in JDBC.
   When pdqProperties is set on the data source of a connection URL, even if the pdqxml file is packaged within the application JAR file, the pdqxml file is ignored. However, if the pdqProperties is not set, the pdqxml file that is in the application JAR file is searched by the runtime (this is the `compatible` mode). If the pdqxml file is found, then the setting is STATIC execution mode. Otherwise, all SQLs are run in JDBC.

Configuring pureQuery to use multiple DB2 package collections:

Set up a pureQuery Java Persistence API (JPA) application to use multiple DB2 package collections.

About this task

It is possible for multiple copies of a database schema to exist. This situation might happen in a partitioned database schema where there is one database for east coast employee data and another database for west coast employee data. In this case, the two databases have the same schema. There might be two databases with two database catalogs. Or there might be only one database, in which case, the high-level qualifier of the table names (the schema name) must be different. Since the schemas are the same, there can be a single set of JPA entities that are used to access both sets of data. There are several ways to configure JPA to handle these situations.

Important: When there are multiple persistence units, either with separate databases or a single database, you must run the wsdbgen command once for each persistence unit.

The following three scenarios exist that require the use of multiple DB2 package collections. If you need more information, read about configuring an application to use IBM Optim PureQuery Runtime.

1. When there are two persistence units with different data source names, using static SQL, two sets of DB2 packages exist: one DB2 package in each database. Since two persistence units exist, two *persistence_unit_name*.pdqxml files for the JPA runtime environment exist.
2. If the tables are in a single database, then two persistence units can also be used. In this case, the data source is the same in both persistence units. However, the schema name property, wsjpa.jdbc.Schema must be different. There are two sets of DB2 packages. Each DB2 package must have a different package name or a different package collection name. Both the wsdb2gen and the DB2 bind command have options to specify the package collection and package names.
3. You can create a single persistence unit, which will eliminate the need to maintain two persistence unit configurations and run the wsdbgen command multiple times. This configuration requires a common package name. Thus the package collection names must be different. Use the createEntityManager(Map map) method, where the map contains the values for the wsjpa.jdbc.Schema and wsjpa.jdbc.CollectionId properties to specify the package collection name and schema name.

Configuring OpenJPA caching to improve performance

The OpenJPA implementation gives you the option of storing frequently used data in the memory to improve performance. OpenJPA provides concurrent data and concurrent query caches that support applications to save persistent object data and query results in memory to share among threads and for use in future queries.

About this task

OpenJPA data cache functionality

The OpenJPA data cache is a cache of persistent object data that operates at the EntityManagerFactory level. This optional-use cache is designed to increase performance while remaining in full compliance with the Java Persistence API (JPA) standard. This means that enabling the caching option can increase the performance of your application, with no changes to your code. The OpenJPA data cache is designed to provide significant performance increases over cacheless operations and ensures that behavior is identical in both cache-enabled and cacheless operations.

When enabled, the cache is examined before accessing the data store. The cache stores data when objects are committed and when persistent objects are loaded from the data store. If operating in a single Java virtual machine (JVM) environment, the JVM maintains and shares a data cache across all EntityManager instances obtained from a particular EntityManagerFactory. The OpenJPA data cache cannot do this in a distributed environment because caches in different JVMs, created from different EntityManagerFactory objects are not synchronized.

Using the OpenJPA cache in a multi-JVM environment can be done by configuring the OpenJPA second level (L2) cache provider plug-in. See the topic, Dynamic cache provider for the JPA 2.0 second level cache, and the section "Using the dynamic cache L2 Cache Provider in a clustered environment", for more information. Configuring the DynaCache plug-in allows for the Data and Query cache content to be replicated and consistent across JVMs. Other alternatives include setting up an event notification framework or using a third-party distributed cache such as IBM WebSphere eXtreme Scale.

Enabling and configuring the OpenJPA data cache

You can enable the OpenJPA data cache for a single or a multiple JVM environment, set its default element size, including soft references, and specify **timeout** values.

To set up and configure the OpenJPA data cache, do the following:

1. To enable the cache for a single JVM, set the openjpa.DataCache property to `true`, and set the openjpa.RemoteCommitProvider property to `sjvm`:

   ```
   <property name="openjpa.DataCache" value="true"/>
   <property name="openjpa.RemoteCommitProvider" value="sjvm"/>
   ```

   To enable the data cache in a distributed environment, the openjpa.RemoteCommitProvider must be configured specifically for the environment, or a third-party cache management utility can be used.
2. The maximum cache size can be adjusted by setting the CacheSize property:

   ```
   <property name="openjpa.DataCache" value="true(CacheSize=5000...
   ```

   By default, the OpenJPA data cache holds 1000 elements. Objects that are pinned into the cache are not counted when determining if the cache size exceeds its maximum size. If the cache overflows, it evicts random elements. You can preserve evicted elements longer with the SoftReferenceSize property. By default, soft references are unlimited. If you must, you can limit the number of soft references or set to 0 to disable soft references completely:

   ```
   <property name="openjpa.DataCache" value="true(CacheSize=5000 SoftReferenceSize=0 ...
   ```
3. You can specify that a cache is cleared at certain times. The EvictionSchedule property of the OpenJPA cache implementation accepts a cron style eviction schedule. The cron format specifies the minute, hour of day, day of month, day of month, and day of the week beginning with 1 for Sunday; the * symbol (asterisk), indicates match all. To schedule a cache to evict at 45 minutes past 3 PM on Sunday every month you would add this property:

   ```
   <property name="openjpa.DataCache" value="true(CacheSize=5000 SoftReferenceSize=0 EvictionSchedule='15,45 * * 1'")/>
   ```
4. You also can specify a cache timeout value for a single class by setting the timeout metadata extension to the amount of time in milliseconds that the data of the class is valid; for example:

   ```
   @Entity
   @DataCache(timeout=10000)
   public class Employee {
       ...
   }
   ```

   Refer to the org.apache.openjpa.persistence.DataCache Javadoc for more information.

After configuring your data cache, you can use it after you restart your application.

Refresh with active DataCache

Refreshing an entity may lead to different behavior with or without a DataCache when a separate process or part of the same application are updated or even deleted the corresponding record in the database. By default, entities are refreshed from the database even when DataCache is active. Therefore, with the default configuration the refresh behaves identically with or without a DataCache. However, a persistence unit can be configured to refresh entities from DataCache with the property openjpa.RefreshFromDataCache for improved performance. Under this configuration, any out-of-band changes that occur in the database record do not appear in the refreshed state of the entity.

Note: Regardless of the **openjpa.RefreshFromDataCache** setting, the DataCache is always bypassed for refresh when locks are active, such as for a pessimistic transaction, in a persistence context. An application may activate **openjpa.RefreshFromDataCache** but can still bypass the DataCache while refreshing an entity by explicitly evicting the entity from DataCache before refresh.

Query Caching functions

OpenJPA provides a concurrent query cache that supports applications to save persistent object data and query results in memory to share among threads and for use in future queries. The query cache stores the object IDs returned by query executions. When you run a query, OpenJPA assembles a key based on the query properties and the parameters used at execution time, and checks for a cached query result. If one

is found, the object IDs in the cached result are looked up, and the resultant persistence-capable objects are returned. Otherwise, the query is executed against the database, and the object IDs loaded by the query are put into the cache.

Configuring or disabling the query cache

You can configure the query cache settings in a similar way to the data cache. The interface provided to the query cache is the org.apache.openjpa.persistence.QueryResultCache class. You can access this class through the OpenJPAEntityManagerFactory.

The default query cache implementation caches 100 query executions in a *least-recently-used* cache. This can be changed by setting the cache size in the CacheSize plug-in property. Like the data cache, the query cache also has a backing soft reference map that can be changed using the SoftReferenceSize property. To keep queries in the cache at all times, you can pin them to a cache. To change the query cache properties do the following:

1. Modify the CacheSize property of the openjpa.QueryCache:
   ```
   <property name="openjpa.QueryCache" value="true("CacheSize=1000, ...
   ```
2. Change the SoftReferenceSize property to enable and control the size of this map:
   ```
   <property name="openjpa.QueryCache" value="true(CacheSize=1000, SoftReferenceSize=100")/>
   ```

   The SoftReferenceSize table is disabled by default. Setting the size enables it.
3. Pin or unpin queries in the cache through the QueryResultCache with this syntax:
   ```
   public void pin(Query q);
   public void unpin(Query q);
   ```

Modifying these properties allows you to make better use of the query cache.

Extending a cache

OpenJPA provides classes that may be extended for further functionality.

- As previously mentioned, if you want to implement a distributed cache that uses an unsupported method for communications, create an implementation of org.apache.openjpa.event.RemoteCommitProvider.
- If you are adding new behavior, extend org.apache.openjpa.datacache.DataCacheImpl.
- To use your own storage mechanism, extend org.apache.openjpa.datacache.AbstractDataCache.
- To add query functionality, you can extend the default org.apache.openjpa.datacache.QueryCacheImpl.
- Implement your own storage mechanism for query results by extending org.apache.openjpa.datacache.AbstractQueryCache

OpenJPA query SQL cache

OpenJPA provides a cache that provides caching of SQL strings used by find operations performed on the entity manager and some queries to manage eagerly fetched relationships. When this cache is enabled, SQL queries used by these operations are generated one time per entity manager factory and can be reused. This cache is enabled by default but can also be configured through the openjpa.jdbc.QuerySQLCache configuration property.

Configuring or disabling the SQL query cache

The query SQL cache can be configured or disabled through the openjpa.jdbc.QuerySQLCache property. By default, this property is set to `true`. When the property is set to `true`, the cache is enabled and uses the org.apache.openjpa.util.CacheMap class for its cache store. The CacheMap is a managed cache, meaning that it limits the number of cache entries and has a cache eviction scheme to manage memory usage. If the cache is set to all the org.apache.openjpa.lib.util.ConcurrentHashMap class is used as a

cache store. The ConcurrentHashMap is not a managed cache so entries remain in the cache for the lifetime of an entity manager factory. This caching mechanism can provide better performance at the expense of increased memory usage. A custom cache store class can also be specified if it implements the java.util.Map interface. To disable the cache, specify the value `false`. See the following examples on how to configure or disable the SQL query cache:

- To use an unmanaged cache:

  ```
  <property name="openjpa.jdbc.QuerySQLCache" value="all"/>
  ```

- To specify a custom cache class:

  ```
  <property name="openjpa.jdbc.QuerySQLCache" value="com.mycompany.MyCustomCache"/>
  ```

- To use an unmanaged cache:

  ```
  <property name="openjpa.jdbc.QuerySQLCache" value="false"/>
  ```

What to do next

You can read more about Caching in the OpenJPA for all caching extensions in the Apache OpenJPA User Guide.

Chapter 10. Administering Internationalization service

This page provides a starting point for finding information about globalization and the internationalization service, a WebSphere extension for improving developer productivity.

With the internationalization service, you can automatically recognize the time zone and location information of the calling client so that your application can act appropriately. The technology enables you to deliver each user, around the world, the right date and time information, the appropriate currencies and languages, and the correct date and decimal formats.

This documentation also includes information about internationalizing interface strings using the localizable-text application programming interface. More introduction...

Task overview: Globalizing applications

An application that can present information to users according to regional cultural conventions is said to be *globalized*: The application can be configured to interact with users from different localities in culturally appropriate ways. In a globalized application, a user in one region sees error messages, output, and interface elements in the requested language. Date and time formats, as well as currencies, are presented appropriately for users in the specified region. A user in another region sees output in the conventional language or format for that region. Globalization consists of two phases: *internationalization* (enabling an application component for multicultural support) and *localization* (translating and implementing a specific regional convention). This product supports globalization through the use of its localizable-text API and internationalization service.

Procedure

- Make sure the server runtime environment is properly configured.

  For more information about supported locales and character encodings, see “Working with locales and character encodings” on page 453.
- Implement message catalogs in your application by using the localizable-text API.

  This product supports the maintenance and deployment of centralized message catalogs for the output of properly formatted, language-specific (*localized*) interface strings.

  For more information about the localizable-text API, see “Task overview: Internationalizing interface strings (localizable-text API)” on page 455.
- Implement more extensive locale support by using the internationalization service.

  With the internationalization service, you can manage the distribution of the internationalization information, or *internationalization context*, that is necessary to support globalized Java Platform, Enterprise Edition (Java EE) application components. Supported application components also include web service client environments and web service-enabled enterprise beans.

  For more information about the internationalization service, see “Task overview: Internationalizing application components (internationalization service)” on page 467.

Globalization

An application that can present information to users according to regional cultural conventions is said to be *globalized*: The application can be configured to interact with users from different localities in culturally appropriate ways. In a globalized application, a user in one region sees error messages, output, and interface elements in the requested language. Date and time formats, as well as currencies, are presented appropriately for users in the specified region. A user in another region sees output in the conventional language or format for that region. Globalization consists of two phases: *internationalization* (enabling an application component for multicultural support) and *localization* (translating and implementing a specific regional convention).

Historically, the creation of globalized applications has been restricted to large corporations writing complex systems. However, given the rise in distributed computing and in the use of the World Wide Web, application developers are pressured to globalize a much wider variety of applications. This trend requires making globalization techniques much more accessible to application developers.

Internationalization of an application is driven by two variables, the time zone and the locale. The *time zone* indicates how to compute the local time as an offset from a standard time like Greenwich Mean Time. The *locale* is a collection of information about language, currency, and the conventions for presenting information like dates. A time zone can cover many locales, and a single locale can span time zones. With both time zone and locale, the date, time, currency, and language for users in a specific region can be determined.

By convention, a given locale is specified with a pair of codes (for language and region) that are governed by different standards. The ISO-639 standard governs the language code; the ISO-3166 standard governs the regional code. In notation, the two codes are typically joined by an underscore (_) character, for example, en_US for English in the United States. In Java code, locales are set and retrieved by means of the java.util.Locale class.

A first step: Localization of interface strings

In an application that is not globalized, the user interface is unalterably written into the application code. Internationalizing a user interface adds a layer of abstraction into the design of an application. The additional layer of abstraction enables you to localize the application for each locale that must be supported by the application.

In a localized application, the locale determines the message catalog from which the application retrieves message strings. Instead of printing an error message, the application represents the error message with some language-neutral information; in the simplest case, each error condition corresponds to a key. To print a usable error message, the application looks up the key in a *message catalog*. Each message catalog is a list of keys with associated strings. Different message catalogs provide strings for the different languages that are supported. The application looks up the key in the appropriate catalog, retrieves the corresponding error message in the requested language, and prints the string for the user.

Localization of text can be used for far more than translating error messages. For example, by using keys to represent each element in a graphical user interface (GUI) and by providing the appropriate message catalogs, the GUI (buttons, menus, and so on) can support multiple languages. Extending support to additional languages requires that you provide message catalogs for those languages; in many cases, the application needs no further modification.

The localizable-text package is a set of Java classes and interfaces that can be used to localize the strings in distributed applications easily. Language-specific string catalogs can be stored centrally so that they can be maintained efficiently.

Globalization challenges in distributed applications

With the advent of Internet-based business computational models, applications increasingly consist of clients and servers that operate in different geographical regions. These differences introduce the following challenges to the task of designing a solid client-server infrastructure:

Clients and servers can run on computers that have different endian architectures or code sets

Clients and servers can reside in computers that have different endian architectures: A client can reside in a little-endian CPU, while the server code runs in a big-endian one. A client might want to call a business method on a server running in a code set different from that of the client.

A client-server infrastructure must define precise endian and code-set tracking and conversion rules. The Java platform has nearly eliminated these problems in a unique way by relying on its

Java virtual machine (JVM), which encodes all of the string data in UCS-2 format and externalizes everything in big-endian format. The JVM uses a set of platform-specific programs for interfacing with the native platform. These programs perform any necessary code set conversions between UCS-2 and the native code set of a platform.

Clients and servers can run on computers with different locale settings

Client and server processes can use different locale settings. For example, a Spanish client might call a business method upon an object that resides on an American English server. Some business methods are locale-sensitive in nature; for example, given a business method that returns a sorted list of strings, the Spanish client expects that list to be sorted according to the Spanish collating sequence, not in the English collating sequence of the server. Because data retrieval and sorting procedures run on the server, the locale of the client must be available to perform a legitimate sort.

A similar consideration applies in instances where the server has to return strings containing date, time, currency, exception messages, and so on, that are formatted according to the cultural expectations of the client.

Clients and servers can reside in different time zones

Client and server processes can run in different time zones. To date, all internationalization literature and resources concentrate mainly on code set and locale-related issues. They have generally ignored the time zone issue, even though business methods can be sensitive to time zone as well as to locale.

For example, suppose that a vendor makes the claim that orders received before 2:00 PM are processed by 5:00 PM the same day. The times given, of course, are in the time zone of the server that is processing the order. It is important to know the time zone of the client to give customers in other time zones the correct times for same-day processing.

Other time zone-sensitive operations include time stamping messages logged to a server, and accessing file or database resources. The concept of Daylight Savings Time further complicates the time zone issue.

Java Platform, Enterprise Edition (Java EE) provides support for application components that run on computers with differing endian architecture and code sets. It does not provide dedicated support for application components that run on computers with different locales or time zones.

The conventional method for solving locale and time zone mismatches across remote application components is to pass one or more extra parameters on all business methods needed to convey the client-side locale or time zone to the server. Although simple, this technique has the following limitations when used in Enterprise JavaBeans (EJB) applications:

- It is intrusive because it requires that one or more parameters be added to all bean methods in the call chain to locale-sensitive or time zone-sensitive methods.
- It is inherently error-prone.
- It is impracticable within applications that do not support modification, such as legacy applications.

The internationalization service addresses the challenges posed by locale and time zone mismatch without incurring the limitations of conventional techniques. The service systematically manages the distribution of internationalization contexts across the various components of EJB applications, including client applications, enterprise beans, and servlets. For more information, see “Task overview: Internationalizing application components (internationalization service)” on page 467.

Working with locales and character encodings

Internationalization support for this product relies on that provided by the Java Platform, Standard Edition (JSE). Support varies by platform.

Procedure

- Verify that the operating system on which the application server is installed supports the locales and encodings that you plan to use.

  Java internationalization support might use underlying services of the operating system. For example, if user IDs for your server are expected to contain non-English characters, make sure that the operating system is configured to process those characters.
- Plan for encoding changes as necessary.

  Consider differences in encoding support among operating system subcomponents. Although this product and the Java platform are based on Unicode encoding, it is not always possible to run applications in a purely Unicode environment.
- Set the console.encoding property as necessary.

Results

If your application produces an UnsupportedEncodingException exception, check your operating system documentation to determine if the target operating system supports the required encoding and adjust the runtime environment as needed. Use the converter.properties file as appropriate to map an unsupported character set to a supported character set. A typical converter.properties file appears below:

```
Shift_JIS=CP943C
EUC-JP=Cp33722C
EUC-JP=Cp33722C
EUC-KR=Cp970
EUC-TW=Cp964
Big5=Co950
GB2312=Cp1386
ISO-2022–KR=ISO2022KR
```

The converter.properties file implements a method for specifying a content type header field that browsers would understand (such as, SHIFT_JIS) and a writer that can output characters correctly (such as, Cp943c).

Language versions offered by this product

This product is offered in several languages, as enabled by the operating platform on which the product is installed.

WebSphere Application Server offers translations for the following languages.

- Brazilian Portuguese
- Chinese (Simplified)
- Chinese (Traditional)
- Czech
- English
- French
- German
- Hungarian
- Italian
- Japanese
- Korean
- Polish
- Romanian
- Russian
- Spanish

Globalization: Resources for learning

Use links in this topic to find relevant supplemental information about globalization. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to this product but is useful all or in part for understanding the product. When possible, links are provided to technical papers and IBM Redbooks® publications that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

View links to additional information about:
- "Programming instructions and examples"
- "Programming specifications"

Programming instructions and examples

- Java internationalization tutorial

  An online tutorial that explains how to use the Java SDK Internationalization API.
- Globalize your On Demand Business

  IBM's portal site for delivering globalized applications.

Programming specifications

- Java 2 Platform Standard Edition 5.0 Development Kit Documentation: Internationalization

  The Java internationalization documentation from Sun Microsystems, including a list of supported locales and encodings. For other versions of the Java platform, click the "Internationalization Home Page" link on that page.
- Java Specification Request 150, Internationalization Service for J2EE

  The specification of the Java internationalization service that was developed through the Java Community Process.
- W3C, Internationalization Core Working Group

  The W3C's Internationalization Core Working Group responsible for investigating the internationalization of web services, in particular, the dependence of web services on language, culture, region, and locale-related contexts.
- Making the WWW truly World Wide

  The W3C effort to make web technologies work with the many writing systems, languages, and cultural conventions of the global community:

Task overview: Internationalizing interface strings (localizable-text API)

This topic summarizes the steps involved in implementing message catalogs through the localizable-text API.

About this task

This product supports the maintenance and deployment of centralized message catalogs for the output of properly formatted, language-specific (*localized*) interface strings.

Procedure

1. Identify localizable text in your application.
2. Create the message catalogs that are necessary for the locales to be supported by your application.
3. In your application code, compose the language-specific strings for output.
4. Using an assembly tool, assemble your application code as one or more application components.
5. Prepare the localizable-text package for deployment with your localized application. In this step, you create a deployment Java archive (JAR) file.

6. Assemble the application modules and the deployment JAR file into a Java Platform, Enterprise Edition (Java EE) application.
7. Deploy and manage the application.

Results

Your application is deployed with localized text.

Identifying localizable text

The first step in localizing strings in an application component is identifying the best candidates for translation.

Procedure

1. Determine which elements of the application need translating. Good candidates for localization include the following:
   - Graphical user interfaces: window titles, menus and menu items, buttons, on-screen instructions
   - Prompts in command-line interfaces
   - Application output: messages and logs
2. Assign a unique key to each element for use in message catalogs for the application. The key provides a language-neutral link between the application and language-specific strings in the message catalogs. Establishing a naming convention for keys before creating the catalogs can make writing code with these keys much more intuitive for interface programmers.

Example

Suppose you are localizing the GUI for a banking system, and the first window contains a pull-down list to use for selecting a type of account.

The labels for the list and the account types in the list are good choices for localization. Three elements require keys: the list and two items in the list.

What to do next

Create message catalogs for the language-specific strings.

Creating message catalogs

Perform this task to begin the localization of strings in an application component.

Before you begin

Identify strings that need to be localized.

About this task

You can create a catalog as either a java.util.ResourceBundle subclass or a Java properties file. The properties-file approach is more common, because properties files can be prepared by people without programming experience and swapped without modifying the application code.

Procedure

1. For each string that is identified for localization, add a line to the message catalog that lists the string key and value in the current language. In a properties file, each line has the following structure:

   ```
   key = string associated with the key
   ```

2. Save the catalog, giving it a locale-specific name. To enable resolution to a specific properties file, the Java API specifies naming conventions for the properties files in a resource bundle as *bundleName_localeID*.properties. Give the set of message catalogs a collective name, for example, BankingResources. For information about locale IDs that are recognized by the Java APIs, see "Resources for learning."

Example

The following English catalog (BankingResources_en.properties) supports the labels for the list and its two list items:

```
accountString = Accounts
savingsString = Savings
checkingString = Checking
```

Do not create compound strings by concatenation (for example, combining the values of savingsString and accountString to form `Savings Accounts` in English. Success depends upon the grammar of the original language (in this case, English) and is not likely to extend to other languages.

The corresponding German catalog (BankingResources_de.properties) supports the labels as follows:

```
accountString = Konten
savingsString = Sparkonto
checkingString = Girokonto
```

What to do next

Write code to compose the language-specific strings.

Composing language-specific strings

Perform this task to complete the localization of strings in an application component.

Before you begin

Create message catalogs for the language-specific strings.

Procedure

1. In application code, create a LocalizableTextFormatter instance, passing in required localization values.
2. Set other localization values as needed for more complex situations.
3. Generate a properly formatted, language-specific string.

What to do next

When the application is finished, deploy your application. For more information, see “Preparing the localizable-text package for deployment” on page 465.

Localization API support

The com.ibm.websphere.i18n.localizabletext package contains classes and interfaces for localizing text.

This package makes extensive use of the internationalization features of the standard Java APIs from Sun Microsystems, including the following classes:

- java.util.Locale

- java.util.TimeZone
- java.util.ResourceBundle
- java.text.MessageFormat

For more information about the standard Java APIs, see “Globalization: Resources for learning” on page 455.

The localizable-text package wraps the Java support and extends it for efficient and simple use in a distributed environment. The primary class used by application programmers is LocalizableTextFormatter. Instances of this class are usually created in server programs, but client programs can also create them. Formatter instances are created for specific resource-bundle names and keys. Client programs that receive a LocalizableTextFormatter instance call its format method. This method uses the locale of the client application to retrieve the appropriate resource bundle and compose a locale-specific message based on the key.

For example, suppose that a distributed application supports both French and English locales; the server is using an English locale and the client, a French locale. The server creates two resource bundles, one each for English and French. When the client makes a request that triggers a message, the server creates a LocalizableTextFormatter instance that contains the name of the resource bundle and the key for the message and passes the instance back to the client.

When the client receives the LocalizableTextFormatter instance, it calls the format method of the object. By using the locale and name of the resource bundle, the format method determines the name of the resource bundle that supports the French locale and retrieves the message that corresponds to the key from the French resource bundle. Formatting of the message is transparent to the client.

In this simple example, the resource bundles reside centrally with the server. They do not have to exist with the client. Part of what the localizable-text package provides is the infrastructure to support centralized catalogs. This implementation uses an enterprise bean (a stateless session bean provided with the localizable-text package) to access the message catalogs. When the client calls the format method on the LocalizableTextFormatter instance, the following events occur:

1. The client application sets the time-zone and locale values in the LocalizableTextFormatter instance, either by passing them explicitly or through default values.
2. A LocalizableTextFormatterEJBFinder call is made to retrieve a reference to the formatter bean.
3. Information from the LocalizableTextFormatter instance, including the time zone and locale of the client, is sent to the formatting bean.
4. The formatting bean uses the name of the resource bundle, the message key, the time zone, and the locale to compose a language-specific message.
5. The formatter bean returns the formatted message to the client.
6. The formatted message is inserted into the LocalizableTextFormatter instance and returned by the format method.

A call to the format method requires at most one remote call, to contact the formatter bean. As an alternative, the LocalizableTextFormatter instance can cache formatted messages, eliminating the remote call for subsequent uses. In addition, you can set a fallback string so that the application can return a readable string even if it cannot access the appropriate message catalog.

The resource bundles can be stored locally. The localizable-text package provides a static variable that indicates whether the bundles are stored locally (LocalizableConfiguration.LOCAL) or remotely (LocalizableConfiguration.REMOTE). However, the setting of this variable applies to all applications running within the same Java virtual machine.

LocalizableTextFormatter class

The LocalizableTextFormatter class, found in the com.ibm.websphere.i18n.localizabletext package, is the primary programming interface for using the localizable-text package. Instances of this class contain the information needed to create language-specific strings from keys and resource bundles.

The LocalizableTextFormatter class extends the java.lang.Object class and implements the following interfaces:

- java.io.Serializable
- com.ibm.websphere.i18n.localizabletext.LocalizableText
- com.ibm.websphere.i18n.localizabletext.LocalizableTextL
- com.ibm.websphere.i18n.localizabletext.LocalizableTextTZ
- com.ibm.websphere.i18n.localizabletext.LocalizableTextLTZ

Creation and initialization of class instances

The LocalizableTextFormatter class supports the following constructors:

- LocalizableTextFormatter()
- LocalizableTextFormatter(String resourceBundleName, String patternKey, String appName)
- LocalizableTextFormatter(String resourceBundleName, String patternKey, String appName, Object[] args)

The LocalizableTextFormatter instance must have certain values, such as a resource-bundle name, a key, and the name of the formatting application. If you do not pass these values in by using the second constructor listed previously, you can set them separately by making the following calls:

- setResourceBundleName(String resourceBundleName)
- setPatternKey(String patternKey)
- setApplicationName(String appName)

You can use a fourth method, setArguments(Object[] args), to set optional localization values after construction. See “Processing of application-specific values” on page 460 at the end of this topic. For a usage example, see “Composing complex strings” on page 462.

API for formatting text

The formatting methods in the LocalizableTextFormatter class generate a string from a set of message keys and resource bundles, based on some combination of locale and time-zone values. Each method corresponds to one of the four localizable-text interfaces implemented. The following list indicates the interface in which each formatting method is defined:

- LocalizableText.format()
- LocalizableTextL.format(java.util.Locale locale)
- LocalizableTextTZ.format(java.util.TimeZone timeZone)
- LocalizableTextLTZ.format(java.util.Locale locale, java.util.TimeZone timeZone)

The format method with no arguments uses the locale and time-zone values set as defaults for the Java virtual machine. All four methods issue LocalizableException objects as needed.

Location of message catalogs and the appName value

Applications written with the localizable-text package can access message catalogs locally or remotely. In a distributed environment, the use of remote, centrally located message catalogs is appropriate. All clients can use the same catalogs, and maintenance of the catalogs is simplified. Local formatting is useful in test situations and appropriate under some circumstances. To support either local or remote formatting, a LocalizableTextFormatter instance must indicate the name of the formatting application.

For example, when an application formats a message by using remote catalogs, the message is actually formatted by an enterprise bean on the server. Although the localizable-text package contains the code to

automate the lookup of the formatter bean and to issue a call to it, the application needs to know the name of the formatter bean. Several methods in the LocalizableTextFormatter class use a value described as *appName*, which refers to the name of the formatting application. It is not necessarily the name of the application in which the value is set.

Caching of messages

LocalizableTextFormatter instances can optionally cache formatted messages so that they do not require reformatting when needed again. By default, caching is not enabled, but you can use a `LocalizableTextFormatter.setCacheSetting(true)` call to enable caching. When caching is enabled and the format method is called, the method determines whether the message is already formatted. If so, the cached message is returned. If the message is not found in the cache, the message is formatted and returned to the caller, and a copy of the message is cached for future use.

If caching is disabled after messages are cached, those messages remain in the cache until the cache is cleared by a call to the LocalizableTextformatter.clearCache method. You can clear the cache at any time; the cache is automatically cleared when any of the following methods is called:
- setResourceBundleName(String resourceBundleName)
- setPatternKey(String patternKey)
- setApplicationName(String appName)
- setArguments(Object[] args)

API for providing fallback information

Under some circumstances, it can be impossible to format a message. The localizable-text package implements a fallback strategy, making it possible to get some information even if a message cannot be formatted correctly into the requested language. The LocalizableTextFormatter instance can optionally store fallback values for a message string, the time zone, and the locale. These values can be ignored unless the LocalizableTextFormatter instance issues an exception. To set fallback values, call the following methods as appropriate:
- setFallBackString(String message)
- setFallBackLocale(Locale locale)
- setFallBackTimeZone(TimeZone timeZone)

For a usage example, see “Generating localized text” on page 464.

Processing of application-specific values

The localizable-text package provides native support for localization based on time zone and locale, but you can construct messages on the basis of other values as well. If you need to consider variables other than locale and time zone in formatting localized text, write your own formatter class.

Your formatter class can extend the LocalizableTextFormatter class or independently implement some or all of the same localizable-text interfaces. As a minimum, your class must implement the java.io.Serializable interface and at least one of the localizable-text interfaces and its corresponding format method. If your class implements more than one localizable-text interface and format method, the order of evaluation of the interfaces is as follows:
1. LocalizableTextLTZ
2. LocalizableTextL
3. LocalizableTextTZ
4. LocalizableText

As an example, the localizable-text package provides a class that reports the time and date (LocalizableTextDateTimeArgument). In that class, date and time formatting is localized in accordance with three values: locale, time zone, and style.

Creating a formatter instance

Perform this task to set localization values for strings in an application component.

About this task

Server programs typically create LocalizableTextFormatter instances that are sent to clients as the result of some operation; clients format the objects at the appropriate time. Less typically, client programs create LocalizableTextFormatter objects locally.

Procedure

1. If needed for your application, write your own formatter class. For more information about implementation, see “LocalizableTextFormatter class” on page 459.
2. In application code, call the appropriate constructor for the formatter class and set required localization values. Some localization values, such as resource bundle name, key and formatting application, must be set, either through a constructor or soon after construction. Other localization values can be set only as needed. For more information about the API, see the related reference.

Example

The following code creates a LocalizableTextFormatter instance by using the default constructor and then sets the required localization values:

```
import com.ibm.websphere.i18n.localizabletext.LocalizableException;
import com.ibm.websphere.i18n.localizabletext.LocalizableTextFormatter;
import java.util.Locale;

public void drawAccountNumberGUI(String accountType) {
   ...
   LocalizableTextFormatter ltf = new LocalizableTextFormatter();
   ltf.setPatternKey("accountNumber");
   ltf.setResourceBundleName("BankingSample.BankingResources");
   ltf.setApplicationName("BankingSample");
   ...
}
```

The line of code in boldface exploits default behavior of the Java platform. By default, the Java platform looks first for a subclass of java.util.ResourceBundle called BankingResources. When none is found, the Java platform looks for a valid properties file of the same name. In this continuing example, a properties file is found.

The application that is requesting a localized message can specify the locale and time zone for message formatting, or the application can use the default values set for the Java virtual machine.

For example, a GUI can enable users to select the language in which to display the interface. A default value must be set initially so that the GUI can be created properly when the application first starts, but users can then change the locale for the GUI to suit their needs. The following code shows how to change the locale used by an application based on the selection of a menu item:

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
...
import java.util.Locale;

public void actionPerformed(ActionEvent event) {
   String action = event.getActionCommand();
   ...
   if (action.equals("en_us")) {
      applicationLocale = new Locale("en", "US");
      ...
   }
   if (action.equals("de_de")) {
```

```
      applicationLocale = new Locale("de", "DE");
      ...
   }
   if (action.equals("fr_fr")) {
      applicationLocale = new Locale("fr", "FR");
      ...
   }
   ...
}
```

For more information, see "Generating localized text."

What to do next

Set optional localization values.

Setting optional localization values

In addition to setting localization values that are required by the LocalizableTextFormatter interface, you can set a number of optional values in application code, either through the constructor or by calling any of several methods for that purpose.

About this task

With optional values, you can do the following actions:
- Compose complex strings from variable substrings
- Customize the formatting of strings, considering variables other than time zone and locale

Procedure

1. In application code, add the optional values into an array of type Object.
   ```
   Object[] arg = {new String(getAccountNumber())};
   ```
2. Pass the array into a LocalizableTextFormatter instance. You can pass the array through the appropriate constructor or call the setArguments(Object[]) method. For a usage example, see “Composing complex strings.”

   Because the array is passed by value rather than by reference, any updates to the array variable after this point are not reflected in the LocalizableTextFormatter instance unless it is reset by calling the setArguments(Object[]) method.

What to do next

Write code to generate the localized text.

Composing complex strings:

Perform this task to insert variable substrings into a localized string.

Before you begin

Identify strings that need to be localized.

About this task

The localized-text package supports the substitution of variable substrings into a localized string that is retrieved from the message catalog by key.

Procedure

1. In the message catalog, specify the location of the substitution in the string to be retrieved. Variable components are designated by braces (for example, {0}).
2. In application code, create a LocalizableTextFormatter instance, passing in an array that contains the variable value. If the variable substring must be localized, you can create a nested LocalizableTextFormatter instance and pass the instance in instead of a value.
3. Generate a localized string. When a format method is called on a formatter instance, the formatter takes each element of the array passed in the previous step and substitutes it for the placeholder with the matching index in the string that is retrieved from the message catalog. For example, the value at index 0 in the array replaces the {0} variable in the retrieved string.

Example

The following line from an English message catalog shows a string with a single substitution:

```
successfulTransaction = The operation on account {0} was successful.
```

The same key in message catalogs for other languages has a translation of this string with the variable at the appropriate location for each language.

The following code shows the creation of a single-element argument array and the creation and use of a LocalizableTextFormatter instance:

```
public void updateAccount(String transactionType) {
   ...
   Object[] arg = {new String(this.accountNumber)};
   ...
   LocalizableTextFormatter successLTF =
      new LocalizableTextFormatter ("BankingResources",
                                    "successfulTransaction",
                                    "BankingSample",
                                    arg);
   ...
   successLTF.format(this.applicationLocale);
   ...
}
```

Nesting formatter instances for localized substrings:

The ability to substitute variable substrings into the strings retrieved from message catalogs adds a level of flexibility to the localizable-text package, but this capability is of limited use unless the variable value can be localized. You can localize this value by nesting LocalizableTextFormatter instances.

Before you begin

Identify strings that need to be localized.

Procedure

1. In the message catalog, add entries that correspond to potential values for the variable substring.
2. In application code, create a LocalizableTextFormatter instance for the variable substring, setting required localization values.
3. Create a LocalizableTextFormatter instance for the primary string, passing in an array that contains the formatter instance for the variable substring.

Example

The following line from an English message catalog shows a string entry with two substitutions and entries to support the localizable variable at index 0 (the second variable in the string, the account number, does not need to be localized):

```
successfulTransaction = The {0} operation on account {1} was successful.
depositOpString = deposit
withdrawOpString = withdrawal
```

The following code shows the creation of the nested formatter instance and its insertion (with the account number variable) into the primary formatter instance:

```
public void updateAccount(String transactionType) {
   ...
   // Successful deposit
   LocalizableTextFormatter opLTF =
      new LocalizableTextFormatter("BankingResources",
                                   "depositOpString",
                "BankingSample");
   Object[] args = {opLTF, new String(this.accountNumber)};
   ...
   LocalizableTextFormatter successLTF =
      new LocalizableTextFormatter ("BankingResources",
                                    "successfulTransaction",
                                    "BankingSample",
                                    args);
   ...
   successLTF.format(this.applicationLocale);
   ...
}
```

Generating localized text

Perform this task to specify the runtime formatting of localized text in an application component.

Before you begin

Create a formatter instance and set the localization values as needed.

Procedure

1. If needed, customize the formatting behavior.
2. In application code, call the appropriate format method.

Example

You can provide fallback behavior for use if the appropriate message catalog is not available at formatting time.

The following code generates a localized string. If the formatting fails, the application retrieves and uses a fallback string instead of the localized string:

```
import com.ibm.websphere.i18n.localizabletext.LocalizableException;
import com.ibm.websphere.i18n.localizabletext.LocalizableTextFormatter;
import java.util.Locale;

public void drawAccountNumberGUI(String accountType){
   ...
   LocalizableTextFormatter ltf = new LocalizableTextFormatter();
   ...
   ltf.setFallBackString("Enter account number: ");
   try {
      msg = new Label(ltf.format(this.applicationLocale), Label.CENTER);
   }
```

```
   catch (LocalizableException le) {
      msg = new Label(ltf.getFallBackString(), Label.CENTER);
   }
   ...
}
```

What to do next

When the application is finished, deploy your application. For more information, see “Preparing the localizable-text package for deployment.”

Customizing the behavior of a formatting method:

Perform this task to change the runtime formatting of localized strings in an application component.

About this task

You can customize formatting behavior by passing your own formatter classes into a LocalizableTextFormatter instance through an array of optional values. This action enables you to consider variables other than locale and time zone when formatting localized text.

Procedure

1. Write your own formatter class. For more information about implementation, see "LocalizableTextFormatter class."
2. In application code, create an instance of your formatter class as appropriate and pass it with any other optional localization values into an instance of LocalizableTextFormatter. When the LocalizableTextFormatter instance reads the instance that has been passed in, it attempts to call the format() method on the passed-in instance. The string returned is then processed with any other elements in the array.

Example

The localizable-text package provides an example of a user-defined class, called LocalizableTextDateTimeArgument. This class enables date and time information to be selectively formatted according to the style values defined in the java.text.DateFormat interface as well as the constants that are defined within the LocalizableTextDateTimeArgument class.

Preparing the localizable-text package for deployment

The LocalizableTextEJBDeploy tool is used to create a deployment Java Archive (JAR) file for the localizable text service. You must deploy the enterprise bean in each enterprise application that requires support for localized text.

Before you begin

Write code to compose the language-specific strings.

Procedure

1. Make sure that the LocalizableTextEJBDeploy tool is included in the class path.

   transition: In versions 6.0.x and earlier, the LocalizableTextEJBDeploy tool used to reside in the file *app_server_root*/lib/ltext.jar. It now resides in the file *app_server_root*/plugins/com.ibm.ws.runtime_1.0.0.jar.
2. Set up a working directory for the LocalizableTextEJBDeploy tool to use. You need to pass this location to the tool through a command-line interface.

3. Run the LocalizableTextEJBDeploy tool. You might be asked if you want to regenerate deployment code for the LocalizableText bean. Do not redeploy the bean; if you do, an incorrect Java Naming and Directory Interface (JNDI) name will be generated.

   To deploy the bean on multiple hosts and servers, run the tool for each host and server combination. This action generates a unique JNDI name for each deployment. After the tool is run, a deployment JAR file is located in the working directory that you specified.

What to do next

Using an assembly tool, assemble the deployment JAR file in an enterprise application with other application components.

As part of preparing for deployment, perform the following:
- Add the resource bundles for your application to the Enterprise Archive (EAR) file as files.
- Add the location of the EAR file to the server class path for the server so that the resource bundles can be located on the virtual host and server.

The same deployment JAR file can be included in several enterprise applications.

LocalizableTextEJBDeploy command

This topic describes the command-line syntax for the LocalizableTextEJBDeploy tool.

transition: In versions 6.0.x and earlier, the LocalizableTextEJBDeploy tool used to reside in the file *app_server_root*/lib/ltext.jar. It now resides in the file *app_server_root*/plugins/com.ibm.ws.runtime_1.0.0.jar.

```
LocalizableTextEJBDeploy
   -a applicationName
   -h virtualHostName
   -i installationDirectory
   -s serverName
   -w workingDirectory
```

Parameters

The required parameters, which can be specified in any order, follow:

applicationName
The name of the formatting session bean. This name is used in LocalizableTextFormatter instances to specify where the actual formatting occurs. If the name cannot be resolved at run time, the format method issues an exception.

virtualHostName
The name of the virtual host on which the formatting session bean is deployed. This value is case-sensitive on all operating platforms.

installationDirectory
The location at which the application server product is installed.

serverName
The name of the application server. If this argument is not specified, the default server name for the product is used.

workingDirectory
A location for the tool to use temporarily.

Task overview: Internationalizing application components (internationalization service)

This topic summarizes the steps involved in using the internationalization service.

About this task

With the internationalization service, you can manage the distribution of the internationalization information, or *internationalization context*, that is necessary to support globalized Java Platform, Enterprise Edition (Java EE) application components. Supported application components also include web service client environments and web service-enabled enterprise beans.

Procedure

1. Use the internationalization context API within application components to obtain or manage internationalization context.

   Servlet and enterprise bean business methods can use internationalization context to perform locale- and time zone-sensitive localizations. Enterprise JavaBeans (EJB) client applications, and server components that are configured to manage internationalization context must use the internationalization context API to set the context elements scoped to their invocations.

   You use the internationalization context API within Web service-enabled Java EE client programs and stateless session beans in the same manner that you would use conventional Java EE application components, with one exception. Internationalization context propagated over Web service requests contains a time zone ID, whereas conventional Remote Method Invocation/ Internet Inter-ORB Protocol (RMI/IIOP) requests propagate complete time zone information, including the raw offset, Daylight Savings Time information, and so on.

2. Assemble internationalized applications.

   The internationalization type specifies the internationalization policy that applies to a servlet or an enterprise bean and, in particular, indicates whether the application component or its hosting Java EE container manages internationalization context. Container internationalization attributes can be specified for container-managed servlet and enterprise bean business methods. These attributes tailor a policy by indicating which context the container scopes to an invocation. Configuring internationalization policies declaratively prescribes, by means of the application deployment descriptor, the distribution and management of context throughout an application.

   As you edit the deployment descriptor for assembly, you can also set the internationalization type and configure any container internationalization attributes for the servlets and enterprise beans in your application.

   You configure internationalization type and container internationalization attributes for Web service-enabled stateless session beans in the same manner as you do for conventional beans.

3. Manage the internationalization service.

   Use the administrative console to enable the service on all application servers.

   By default, the service is enabled within Java EE client environments but is disabled on application servers. You must enable the service on all application servers hosting your servlets and enterprise beans to use internationalization context.

4. Troubleshoot the internationalization service as needed.

   Use the administrative console to enable the trace service to log internationalization service messages when debugging your applications.

   The trace strings for the internationalization service follow; use both:

   ```
   com.ibm.ws.i18n.context.*=all=enabled:com.ibm.websphere.i18n.context.*=all=enabled
   ```

Internationalization service

In a distributed client-server environment, application processes can run on different machines, configured for different locales, corresponding to different cultural conventions; they can also be located across geographical boundaries. The internationalization service can help manage your application in a globally distributed environment.

For an understanding of how differences in locale impact application development, read “Globalization” on page 451.

Java Platform, Enterprise Edition (Java EE) provides support for application components that run on computers with differing endian architecture and code sets. It does not provide dedicated support for application components that run on computers with different locales or time zones.

The internationalization service addresses the challenges posed by locale and time zone mismatch without incurring the limitations of conventional techniques. The service systematically manages the distribution of internationalization contexts across the various components of EJB applications, including client applications, enterprise beans, and servlets.

The service works by associating an internationalization context with every service request within an application. When a client-side component calls a business method, the internationalization service interposes by obtaining the internationalization context associated with the current client-side process and by attaching that context to the outgoing request. On the server side, the internationalization service again interposes by detaching the context from the incoming request and associating it with the server-side process on which the business method will run, effectively scoping the context to the business method. For HTTP requests, the caller context is constructed from the HTTP attributes and default values. The service propagates internationalization context on subsequent business method invocations in the same manner, which distributes the context of the originating request over the entire chain of business method invocations.

This basic operation of scoping and propagation is defined precisely by *internationalization context management policies*. Internationalization policies specify whether an application component or its hosting Java EE container are to manage internationalization context. For container-managed components, the policy indicates which internationalization context the container scopes to invocations on that component. Server components configured to manage internationalization context, as well as EJB clients, must use the internationalization context API to manage the internationalization context elements scoped to their invocations.

Every application component has a default policy, which can be overridden and tailored for servlets and enterprise beans at assembly time.

At run time, application components can use the internationalization context API to get any element of the internationalization contexts scoped to an invocation. To programmatically access context elements, application components first resolve an internationalization context API reference, then call the appropriate API method to access the various context elements, such as the caller locale or the invocation time zone. These elements can be used in calls to Java SDK internationalization API methods; for example, to perform localizations such as formatting messages, configuring dates, or comparing strings.

Assembling internationalized applications

Perform this task to configure application components for deployment with the internationalization service.

About this task

Use an assembly tool to configure internationalization in the deployment descriptors for servlets and enterprise beans.

Procedure

1. Set the **internationalization type**.

   All servlets and enterprise beans have an internationalization type setting that specifies whether internationalization context is managed by the application component or by its hosting Java Platform, Enterprise Edition (Java EE) container during invocations of their respective life cycle and business methods. The internationalization type can be configured for all server application components except entity beans, which are container-managed only.

   By default, all server components use container-managed internationalization (CMI). The default setting suffices in most cases; when it does not, modify the internationalization type setting by completing the steps that are described in one of the following topics:
   - "Setting the internationalization type for servlets"
   - "Setting the internationalization type for enterprise beans" on page 471

2. Set the **container internationalization attribute**.

   You can associate CMI servlets, and business methods of CMI enterprise beans, with a container internationalization attribute. That attribute specifies which of three internationalization contexts (**Caller**, **Server**, or **Specified**) the container is to scope to an invocation. When running as specified, the container internationalization attribute also specifies the custom internationalization context elements.

   Named container internationalization attributes can be associated with sets of servlets or with sets of Enterprise JavaBeans (EJB) business methods. Initially, CMI servlets and business methods implicitly run as caller and do not associate with a container internationalization attribute. When the implicit behavior or an associated attribute setting is unsuitable, configure an attribute by completing the steps that are described in one of the following topics:
   - "Configuring container internationalization for servlets" on page 470
   - "Configuring container internationalization for enterprise beans" on page 471

Setting the internationalization type for servlets

This task sets the internationalization type for a servlet within a Web module.

Before you begin

This topic assumes that you have an assembly tool such as Rational Application Developer.

For information about assembly, refer to the documentation for your assembly tool. The steps in this topic refer to Rational Application Developer.

This topic assumes that you have started the assembly tool, configured the assembly tool for work on Java Platform, Enterprise Edition (Java EE) modules, and created or imported a dynamic Web project.

Procedure

1. In the Java EE perspective, open the Web project for which you want to set the internationalization type.
   a. Double-click **Dynamic Web Projects**.
   b. Double-click the name of the Web project to see its contents.
   c. Double-click the deployment descriptor object.

   The Web Deployment Descriptor panel is displayed.
2. In the Web Deployment Descriptor panel, click the Servlets tab.
3. Scroll down to **WebSphere Programming Model Extensions** and then **Internationalization**.
4. From the **Servlets and JSPs** list of the Servlets panel, select the servlet for which you want to set the internationalization type.
5. Under **Internationalization**, select a value from the **Internationalization type** list. Valid values are `Application` or `Container`.
6. From the menu bar, click **File > Save**.

Results

The internationalization type setting is assigned to the servlet.

What to do next

If you selected container-managed internationalization, you can then set container-managed internationalization attributes for methods within the servlet. For more information, see "Configuring container internationalization for servlets."

Configuring container internationalization for servlets

This task configures container internationalization for a servlet within a Web module.

Before you begin

This topic assumes that you have an assembly tool such as Rational Application Developer.

For information about assembly, refer to the documentation for your assembly tool. The steps in this topic refer to Rational Application Developer.

This topic assumes that you have started the assembly tool, configured the assembly tool for work on Java Platform, Enterprise Edition (Java EE) modules, and created or imported a dynamic Web project.

You must also have set the internationalization type of one or more servlets in a Web project to `Container`.

About this task

This procedure relates one or more servlets to a container-managed internationalization attribute.

Procedure

1. In the Java EE perspective, open the Web project for which you want to configure container internationalization.
   a. Double-click **Dynamic Web Projects**.
   b. Double-click the name of the Web project to see its contents.
   c. Double-click the deployment descriptor object.

   The Web Deployment Descriptor panel is displayed.
2. In the Web Deployment Descriptor panel, click the Servlets tab.
3. Scroll down to **WebSphere Programming Model Extensions** and then **Internationalization**.
4. Following **Container-managed Internationalization Attribute**, set the **Run As** field by selecting `Caller`, `Server`, or `Specified`.
5. If the servlet is to be run as Specified, select an internationalization policy from the **Specified** list or define a new policy.
   a. To define an internationalization policy, click **New**. The New Specified Initialization wizard is displayed.
   b. In the **Description** field, give the policy a name.
   c. If needed, set a time zone ID and add a time zone description. If you do not find the appropriate time zone in the ID list, click **Customize** to define one relative to Greenwich Mean Time (GMT).
   d. Create at least one locale for the policy. To create a locale, click **Add**; select a language and (optional) geographic region; specify a variant as needed. Add a locale description and click **OK** to finish. The new locale is added to the **Locales** list.
   e. If more than one locale is defined for the policy, select a locale from the **Locales** list and click **Finish**. Otherwise, just click **Finish**

6. From the menu bar, click **File > Save**.

Results

Selected servlets are now configured to run under the associated internationalization settings.

Setting the internationalization type for enterprise beans

This task sets the internationalization type for an enterprise bean within an Enterprise JavaBeans (EJB) module.

Before you begin

This topic assumes that you have an assembly tool such as Rational Application Developer.

For information about assembly, refer to the documentation for your assembly tool. The steps in this topic refer to Rational Application Developer.

This topic assumes that you have started the assembly tool, configured the assembly tool for work on Java Platform, Enterprise Edition (Java EE) modules, and created or imported an EJB project.

About this task

Container-managed internationalization (CMI) is the default type; entity beans cannot be set to application-managed internationalization (AMI). Use CMI also for stateless session beans that are enabled for Web services.

Procedure

1. In the Java EE perspective, open the EJB project for which you want to set the internationalization type.
   a. Double-click **EJB Projects**.
   b. Double-click the name of the EJB project to see its contents.
   c. Double-click the deployment descriptor object.

   The EJB Deployment Descriptor panel is displayed.
2. In the EJB Deployment Descriptor panel, click the Internationalization tab. Any enterprise beans that are already configured for AMI are displayed in the **Internationalization type** list.
3. To set the internationalization type for any other enterprise beans to AMI, click **Add** following the **Internationalization type** list. The Internationalization Type wizard opens. Only message-driven or session beans can be selected.
4. Select the beans that you want to set and click **Finish** to exit the wizard.
5. From the menu bar, click **File > Save**.

Results

The internationalization type is assigned to the bean.

What to do next

For beans that use container-managed internationalization, you can then set container-managed internationalization attributes. For more information, see "Configuring container internationalization for enterprise beans."

Configuring container internationalization for enterprise beans

This task configures container internationalization for enterprise bean business methods.

Before you begin

This topic assumes that you have an assembly tool such as Rational Application Developer.

For information about assembly, refer to the documentation for your assembly tool. The steps in this topic refer to Rational Application Developer.

This topic assumes that you have started the assembly tool, configured the assembly tool for work on Java Platform, Enterprise Edition (Java EE) modules, and created or imported an EJB project.

You must also have one or more enterprise beans set to container-managed internationalization (CMI) by default.

About this task

This procedure relates one or more business methods to one or more container-managed internationalization (CMI) attributes. Use this procedure also for stateless session beans that are enabled for Web services.

Procedure

1. In the Java EE perspective, open the EJB project for which you want to configure container internationalization.
   a. Double-click **EJB Projects**.
   b. Double-click the name of the EJB project to see its contents.
   c. Double-click the deployment descriptor object.

   The EJB Deployment Descriptor panel is displayed.
2. In the EJB Deployment Descriptor panel, click the Internationalization tab. Any business methods that are already configured are displayed in the **Internationalization attributes** list.
3. To configure a CMI business method, click **Add** following the **Internationalization attributes** list. The Internationalization Attributes wizard opens.
4. Set the **Run As** field by selecting `Caller`, `Server`, or `Specified`. Add a meaningful description. As a group, the CMI attribute settings comprise an internationalization policy.
   - The description appears as `Internationalization` *`description`* (*`runAsSetting`*) in the **Internationalization attributes** list when you are finished.
   - If you do not provide a description, the policy name appears as `Internationalization` (*`runAsSetting`*).

   If the bean is to be run as Specified, complete the following steps to specify the context elements that the container scopes to bean method invocations.
   a. Set a time zone ID and add a time zone description as needed. If you do not find the appropriate time zone in the ID list, click **Custom** to define one relative to Greenwich Mean Time (GMT).
   b. Set a locale. Select a language and (optional) geographic region; specify a variant as needed. Add a locale description as needed and click **OK** to finish.
5. Click **Next**.
6. Select the beans for which you want to configure method-level internationalization attributes and click **Next**.
7. Select the methods that you want to configure and click **Next**. A check box is displayed next to each method name that you select.
   - Click **Apply to All** to place a check box next to the displayed bean name.
   - Click **Select Beans** to select more beans with CMI.
8. Click **Finish** to exit the wizard.
9. From the menu bar, click **File > Save**.

Results

The bean methods are now configured to run under the associated internationalization settings.

Using the internationalization context API

Enterprise JavaBeans (EJB) client applications, servlets, and enterprise beans can programmatically obtain and manage internationalization context using the internationalization context API. For Web service client applications, you use the API to obtain and manage internationalization context in the same manner as for EJB clients.

Before you begin

The java.util and com.ibm.websphere.i18n.context packages contain all of the classes necessary to use the internationalization service within an EJB application.

Procedure

1. Gain access to the internationalization context API.

   Resolve internationalization context API references once over the life cycle of an application component, within the initialization method of that component (for example, within the init method of servlets, or within the SetXxxContext method of enterprise beans). For Web service client programs, resolve a reference to the internationalization context API during initialization. For stateless session beans enabled for Web services, resolve the reference in the setSessionContext method.
2. Access caller locales and time zones.

   Every remote invocation of an application component has an associated caller internationalization context associated with the thread that is running that invocation. A caller context is propagated by the internationalization service and middleware to the target of a request, such as an Enterprise JavaBeans (EJB) business method or servlet service method. This task also applies to Web service client programs.
3. Access invocation locales and time zones.

   Every remote invocation of a servlet service or Enterprise JavaBeans (EJB) business method has an invocation internationalization context associated with the thread that is running that invocation. Invocation context is the internationalization context under which servlet and business method implementations run; it is propagated on subsequent invocations by the internationalization service and middleware. This task also applies to Web service client programs.

Results

The resulting components are said to use *application-managed internationalization* (AMI). For more information about AMI, see “Internationalization context: Management policies” on page 488.

Example

Each supported application component uses the internationalization context API differently. Code examples are provided that illustrate how to use the API within each component type. Differences in API usage, as well as other coding tips, are noted in comments that precede the relevant statement blocks.

- Managing internationalization context in an EJB client program
- Managing internationalization context in a servlet
- Managing internationalization context in a session bean
- Representing internationalization context in a SOAP header

Managing internationalization context in an EJB client program: The following code example illustrates how to use the internationalization context API within a contained EJB client program or Web service client program.

```
//----------------------------------------
// Basic Example: J2EE EJB client.
//----------------------------------------
package examples.basic;

//----------------------------------------
// INTERNATIONALIZATION SERVICE: Imports.
//----------------------------------------
import com.ibm.websphere.i18n.context.UserInternationalization;
import com.ibm.websphere.i18n.context.Internationalization;
import com.ibm.websphere.i18n.context.InvocationInternationalization;

import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import java.util.Locale;
import java.util.SimpleTimeZone;

public class EjbClient {

  public static void main(String args[]) {

    //------------------------------------------------
    // INTERNATIONALIZATION SERVICE: API references.
    //------------------------------------------------
    UserInternationalization userI18n = null;
    Internationalization callerI18n = null;
    InvocationInternationalization invocationI18n = null;

    //------------------------------------------------
    // INTERNATIONALIZATION SERVICE: JNDI name.
    //------------------------------------------------
    final String UserI18NUrl =
        "java:comp/websphere/UserInternationalization";

    //------------------------------------------------
    // INTERNATIONALIZATION SERVICE: Resolve the API.
    //------------------------------------------------
    try {
     Context initialContext = new InitialContext();
     userI18n = (UserInternationalization)initialContext.lookup(
         UserI18NUrl);
     callerI18n = userI18n.getCallerInternationalization();
     invI18n = userI18n.getInvocationInternationalization ();
    } catch (NamingException ne) {
      log("Error: Cannot resolve UserInternationalization: Exception: " + ne);
    } catch (IllegalStateException ise) {
      log("Error: UserInternationalization is not available: " + ise);
    }
    ...

    //--------------------------------------------------------------------
    // INTERNATIONALIZATION SERVICE: Set invocation context.
    //
    // Under Application-managed Internationalization (AMI), contained EJB
    // client programs may set invocation context elements. The following
    // statements associate the supplied invocation locale and time zone
    // with the current thread. Subsequent remote bean method calls will
    // propagate these context elements.
    //--------------------------------------------------------------------
    try {
      invocationI18n.setLocale(new Locale("fr", "FR", ""));
      invocationI18n.setTimeZone("ECT");
    } catch (IllegalStateException ise) {
      log("An anomaly occurred accessing Invocation context: " + ise );
    }
    ...
```

```
    //---------------------------------------------------------------
    // INTERNATIONALIZATION SERVICE: Get locale and time zone.
    //
    // Under AMI, contained EJB client programs can get caller and
    // invocation context elements associated with the current thread.
    // The next four statements return the invocation locale and time zone
    // associated above, and the caller locale and time zone associated
    // internally by the service. Getting a caller context element within
    // a contained client results in the default element of the JVM.
    //---------------------------------------------------------------
    Locale invocationLocale = null;
    SimpleTimeZone invocationTimeZone = null;
    Locale callerLocale = null;
    SimpleTimeZone callerTimeZone = null;
    try {
      invocationLocale = invocationI18n.getLocale();
      invocationTimeZone =
          (SimpleTimeZone)invocationI18n.getTimeZone();
      callerLocale = callerI18n.getLocale();
      callerTimeZone = SimpleTimeZone)callerI18n.getTimeZone();
    } catch (IllegalStateException ise) {
      log("An anomaly occurred accessing I18n context: " + ise );
    }

    ...
  } // main

  ...
  void log(String s) {
    System.out.println (((s == null) ? "null" : s));
  }
} // EjbClient
```

Managing internationalization context in a servlet: The following code example illustrates how to use the internationalization context API within a servlet. Note comments in the init and doPost methods.

```
...
//---------------------------------------------------------------
// INTERNATIONALIZATION SERVICE: Imports.
//---------------------------------------------------------------
import com.ibm.websphere.i18n.context.UserInternationalization;
import com.ibm.websphere.i18n.context.Internationalization;
import com.ibm.websphere.i18n.context.InvocationInternationalization;

import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import java.util.Locale;

public class J2eeServlet extends HttpServlet {

  ...
  //---------------------------------------------------------------
  // INTERNATIONALIZATION SERVICE: API references.
  //---------------------------------------------------------------
  protected UserInternationalization userI18n = null;
  protected Internationalization i18n = null;
  protected InvocationInternationalization invI18n = null;

  //-------------------------------------------------------------
  // INTERNATIONALIZATION SERVICE: JNDI name.
  //-------------------------------------------------------------
  public static final String UserI18NUrl =
      "java:comp/websphere/UserInternationalization";

  protected Locale callerLocale = null;
```

```
protected Locale invocationLocale = null;

/**
 * Initialize this servlet.
 * Resolve references to the JNDI initial context and the
 * internationalization context API.
 */
public void init() throws ServletException {

  //-------------------------------------------------------------------
  // INTERNATIONALIZATION SERVICE: Resolve API.
  //
  // Under Container-managed Internationalization (CMI), servlets have
  // read-only access to invocation context elements. Attempts to set these
  // elements result in an IllegalStateException.
  //
  // Suggestion: cache all internationalization context API references
  // once, during initialization, and use them throughout the servlet
  // lifecycle.
  //-------------------------------------------------------------------
  try {
    Context initialContext = new InitialContext();
    userI18n = (UserInternationalization)initialContext.lookup(UserI18nUrl);
    callerI18n = userI18n.getCallerInternationalization();
    invI18n = userI18n.getInvocationInternationalization();
  } catch (NamingException ne) {
    throw new ServletException("Cannot resolve UserInternationalization" + ne);
  } catch (IllegalStateException ise) {
    throw new ServletException ("Error: UserInternationalization is not
       available: " + ise);
  }
  ...
} // init

/**
 * Process incoming HTTP GET requests.
 * @param request Object that encapsulates the request to the servlet
 * @param response Object that encapsulates the response from the
 *    Servlet.
 */
public void doGet(
    HttpServletRequest  request,
    HttpServletResponse response)
  throws ServletException, IOException {
  doPost(request, response);
} // doGet

/**
 * Process incoming HTTP POST requests
 * @param request Object that encapsulates the request to
 *    the Servlet.
 * @param response Object that encapsulates the response from
 *    the Servlet.
 */
public void doPost(
    HttpServletRequest  request,
    HttpServletResponse response)
  throws ServletException, IOException {

  ...
  //-------------------------------------------------------------------
  // INTERNATIONALIZATION SERVICE: Get caller context.
  //
  // The internationalization service extracts the accept-languages
  // propagated in the HTTP request and associates them with the
  // current thread as a list of locales within the caller context.
  // These locales are accessible within HTTP Servlet service methods
```

```
    // using the caller internationalization object.
    //
    // If the incoming HTTP request does not contain accept languages,
    // the service associates the server's default locale. The service
    // always associates the GMT time zone.
    //
    //---------------------------------------------------------------------
    try {
      callerLocale = callerI18n.getLocale();   // caller locale
      // the following code enables you to get invocation locale,
      // which depends on the Internationalization policies.
      invocationLocale = invI18n.getLocale();  // invocation locale
    } catch (IllegalStateException ise) {
      log("An anomaly occurred accessing Invocation context: " + ise);
    }
    // NOTE: Browsers may propagate accept-languages that contain a
    // language code, but lack a country code, like "fr" to indicate
    // "French as spoken in France."  The following code supplies a
    // default country code in such cases.
    if (callerLocale.getCountry().equals(""))
      callerLocale = AccInfoJBean.getCompleteLocale(callerLocale);

    // Use iLocale in JDK locale-sensitive operations, etc.
    ...
  } // doPost

  ...
  void log(String s) {
    System.out.println (((s == null) ? "null" : s));
  }
} // CLASS J2eeServlet
```

Managing internationalization context in a session bean: This code example illustrates how to perform a localized operation using the internationalization service within a session bean or Web service-enabled session bean.

```
...
//-------------------------------------------------------------
// INTERNATIONALIZATION SERVICE: Imports.
//-------------------------------------------------------------
import com.ibm.websphere.i18n.context.UserInternationalization;
import com.ibm.websphere.i18n.context.Internationalization;
import com.ibm.websphere.i18n.context.InvocationInternationalization;

import javax.naming.InitialContext;
import javax.naming.Context;
import javax.naming.NamingException;
import java.util.Locale;

/**
 * This is a stateless Session Bean Class
 */
public class J2EESessionBean implements SessionBean {

  //-------------------------------------------------------------
  // INTERNATIONALIZATION SERVICE: API references.
  //-------------------------------------------------------------
  protected UserInternationalization        userI18n = null;
  protected InvocationInternationalization  invI18n  = null;

  //-------------------------------------------------------------
  // INTERNATIONALIZATION SERVICE: JNDI name.
  //-------------------------------------------------------------
  public static final String UserI18NUrl =
      "java:comp/websphere/UserInternationalization";
  ...
```

```
  /**
   * Obtain the appropriate internationalization interface
   * reference in this method.
   * @param ctx javax.ejb.SessionContext
   */
  public void setSessionContext(javax.ejb.SessionContext ctx) {

    //-------------------------------------------------------------
    // INTERNATIONALIZATION SERVICE: Resolve the API.
    //-------------------------------------------------------------
    try {
      Context initialContext = new InitialContext();
      userI18n = (UserInternationalization)initialContext.lookup(
          UserI18NUrl);
      invI18n = userI18n.getInvocationInternationalization();
    } catch (NamingException ne) {
      log("Error: Cannot resolve UserInternationalization: Exception: " + ne);

    } catch (IllegalStateException ise) {
      log("Error: UserInternationalization is not available: " + ise);
    }
  } // setSessionContext

 /**
  * Set up resource bundle using I18n Service
  */
  public void setResourceBundle()
  {
    Locale invLocale   = null;

    //-------------------------------------------------------------
    // INTERNATIONALIZATION SERVICE: Get invocation context.
    //-------------------------------------------------------------
    try {
      invLocale = invI18n.getLocale();
    } catch (IllegalStateException ise) {
      log ("An anomaly occurred while accessing Invocation context: " + ise );
    }
    try {
      Resources.setResourceBundle(invLocale);
      // Class Resources provides support for retrieving messages from
      // the resource bundle(s). See Currency Exchange sample source code.
    } catch (Exception e) {
      log("Error: Exception occurred while setting resource bundle: " + e);
    }
  } // setResourceBundle

  /**
   * Pass message keys to get the localized texts
   * @return java.lang.String []
   * @param key java.lang.String []
   */
  public String[] getMsgs(String[] key) {
    setResourceBundle();
    return Resources.getMsgs(key);
  }

  ...
  void log(String s) {
    System.out.println(((s == null) ? ";null" : s));
  }
} // CLASS J2EESessionBean
```

Representing internationalization context in a SOAP header: This code example illustrates how internationalization context is represented within the SOAP header of a Web service request.

```
<InternationalizationContext>
   <Locales>
      <Locale>
         <LanguageCode>ja</LanguageCode>
         <CountryCode>JP</CountryCode>
         <VariantCode>Nihonbushi</VariantCode>
      </Locale>
      <Locale>
         <LanguageCode>fr</LanguageCode>
         <CountryCode>FR</CountryCode>
      </Locale>
      <Locale>
         <LanguageCode>en</LanguageCode>
         <CountryCode>US</CountryCode>
      </Locale>
   </Locales>
   <TimeZoneID>JST</TimeZoneID>
</InternationalizationContext>
```

This representation is valid against the following schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="InternationalizationContext"
                 type="InternationalizationContextType">
    </xsd:element>

    <xsd:complexType name="InternationalizationContextType">
        <xsd:sequence>
            <xsd:element name="Locales"
                         type="LocalesType">
            </xsd:element>
            <xsd:element name="TimeZoneID"
                         type="xsd:string">
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="LocalesType">
        <xsd:sequence>
            <xsd:element name="Locale"
                         type="LocaleType"
                         minOccurs="0"
                         maxOccurs="unbounded">
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>

    <xsd:complexType name="LocaleType">
        <xsd:sequence>
            <xsd:element name="LanguageCode"
                         type="xsd:string"
                         minOccurs="0"
                         maxOccurs="1">
            </xsd:element>
            <xsd:element name="CountryCode"
                         type="xsd:string"
                         minOccurs="0"
                         maxOccurs="1">
            </xsd:element>
            <xsd:element name="VariantCode"
                         type="xsd:string"
                         minOccurs="0"
                         maxOccurs="1">
            </xsd:element>
```

```
        </xsd:sequence>
    </xsd:complexType>

</xsd:schema>
```

Gaining access to the internationalization context API

Perform this task to access the internationalization service by resolving a reference to the internationalization context API.

About this task

Resolve internationalization context API references once over the life cycle of an application component, within the initialization method of that component (for example, within the init method of servlets, or within the SetXxxContext method of enterprise beans). For Web service client programs, resolve a reference to the internationalization context API during initialization. For stateless session beans enabled for Web services, resolve the reference in the setSessionContext method.

Procedure

1. Resolve a reference to the UserInternationalization interface by performing a lookup on the Java Naming and Directory Interface (JNDI) name `java:comp/websphere/UserInternationalization`. For example:

```
//--------------------------------------------------------------------
// Internationalization context imports.
//--------------------------------------------------------------------
import com.ibm.websphere.i18n.context.*;
import javax.naming.*;
...

public class MyApplication {
  ...

  //--------------------------------------------------------------------
  // Resolve a reference to the UserInternationalization interface.
  //--------------------------------------------------------------------
  InitialContext initCtx = null;
  UserInternationalization userI18n = null;
  final String UserI18nUrl = "java:comp/websphere/UserInternationalization";
  try {
    initCtx = new InitialContext();
    userI18n = (UserInternationalization)initCtx.lookup(UserI18nUrl);
  }
  catch (NamingException ne) {
    // UserInternationalization URL is unavailable.
  }
```

   If the UserInternationalization object is unavailable because of an anomaly or a restriction, the JNDI lookup invocation issues a javax.naming.NameNotFoundException exception that contains the java.lang.IllegalStateException instance.

2. Use the UserInternationalization reference to create references to the CallerInternationalization or InvocationInternationalization objects, which provide access to elements of the Caller or Invocation internationalization contexts, respectively. The CallerInternationalization reference can be bound to the Internationalization interface only; the InvocationInternationalization reference can be bound to either the Internationalization or the InvocationInternationalization interfaces, depending on whether the application requires read-only or read-write access to the invocation context. For example:

```
  ...
  //--------------------------------------------------------------------
  // Resolve references to the Internationalization and
  // InvocationInternationalization interfaces.
  //--------------------------------------------------------------------
  Internationalization callerI18n = null;
  InvocationInternationalization invocationI18n = null;
```

```
try {
  callerI18n = userI18n.getCallerInternationalization();
  invocationI18n = userI18n.getInvocationInternationalization();
}
catch (IllegalStateException ise) {
  // An Internationalization interface(s) is unavailable.
}
```

Accessing caller locales and time zones

Perform this task to access elements of the caller internationalization context.

Before you begin

An application component must first resolve a reference to the CallerInternationalization object and then bind it to the Internationalization interface.

About this task

Every remote invocation of an application component has an associated caller internationalization context associated with the thread that is running that invocation. A caller context is propagated by the internationalization service and middleware to the target of a request, such as an Enterprise JavaBeans (EJB) business method or servlet service method. This task also applies to Web service client programs.

Procedure

1. Obtain the desired caller context elements.

   ```
   java.util.Locale [] myLocales  = null;
   try {
     myLocales  = callerI18n.getLocales();
   }
   catch (IllegalStateException ise) {
     // The Caller context is unavailable;
     // is the service started and enabled?
   }
   ...
   ```

   The Internationalization interface contains the following methods to get caller internationalization context elements:
   - **Locale [] getLocales()** Returns the list of caller locales that are associated with the current thread.
   - **Locale getLocale()** Returns the first in the list of caller locales that are associated with the current thread.
   - **TimeZone getTimeZone()** Returns the SimpleTimeZone caller that is associated with the current thread.

   The Internationalization interface supports read-only access to internationalization context within application components. Methods of the Internationalization interface are available to all EJB application components and are used in the same manner for each, but the method semantics vary according to the component type. For instance, when obtaining the caller locale within an EJB client application, the interface returns the default locale of the host Java virtual machine (JVM); in contrast, when obtaining caller context within a servlet service method (for example, doPost or doGet methods), the interface returns the first locale (accept-language) propagated within the corresponding HTML request. See Internationalization context for a discussion of how the service propagates internationalization context throughout an application.
2. Use the caller context elements to localize computations under a locale or time zone of the calling process.

   ```
   DateFormat df = DateFormat.getDateInstance(myLocale);
   String localizedDate = df.getDateInstance().format(aDateInstance);
   ...
   ```

Accessing invocation locales and time zones

Perform this task to access elements of the invocation internationalization context.

Before you begin

An application component must first resolve a reference to the InvocationInternationalization object and then bind it to the InvocationInternationalization interface of the internationalization context API.

About this task

Every remote invocation of a servlet service or Enterprise JavaBeans (EJB) business method has an invocation internationalization context associated with the thread that is running that invocation. Invocation context is the internationalization context under which servlet and business method implementations run; it is propagated on subsequent invocations by the internationalization service and middleware. This task also applies to Web service client programs.

Procedure

1. Obtain the desired invocation context elements.

   ```
   java.util.Locale myLocale;
   try {
     myLocale = invocationI18n.getLocale();
   }
   catch (IllegalStateException ise) {
     // The invocation context is unavailable;
     // is the service started and enabled?
   }
   ...
   ```

   The InvocationInternationalization interface contains the following methods to both get and set invocation internationalization context elements:
   - **Locale [] getLocales()**. Returns the list of invocation locales that is associated with the current thread.
   - **Locale getLocale()**. Returns the first in the list of invocation locales that is associated with the current thread.
   - **TimeZone getTimeZone()**. Returns the SimpleTimeZone invocation that is associated with the current thread.
   - **setLocales(Locale [])**. Sets the list of invocation locales that are associated with the current thread to the supplied list.
   - **setLocale(Locale)**. Sets the list of invocation locales that are associated with the current thread to a list that contains the supplied locale.
   - **setTimeZone(TimeZone)**. Sets the invocation time zone that is associated with the current thread to the supplied SimpleTimeZone.
   - **setTimeZone(String)**. Sets the invocation time zone that is associated with the current thread to a SimpleTimeZone that has the supplied ID.

   The InvocationInternationalization interface supports read and write access to invocation internationalization context within application components. However, according to internationalization context management policies, only components configured to manage internationalization context (application-managed internationalization, or AMI, components) have write access to invocation internationalization context elements. Calls to set invocation context elements within container-managed internationalization (CMI) application components result in a java.lang.IllegalStateException exception. Any differences in how application components can use InvocationInternationalization methods are explained in Internationalization context.
2. Use the invocation context elements to localize a computation under a locale or time zone of the calling process.

   ```
   DateFormat df = DateFormat.getDateInstance(myLocale);
     String localizedDate = df.getDateInstance().format(aDateInstance);
     ...
   ```

Example

In the following code example, locale (en,GB) and simple time zone (GMT) transparently propagate on the call to the myBusinessMethod method. Server-side application components, such as `myEjb`, can use the InvocationInternationalization interface to obtain these context elements.

```
...
//--------------------------------------------------------------------
// Set the invocation context under which the business method or
// servlet will run and propagate on subsequent remote business
// method invocations.
//--------------------------------------------------------------------
try {
  invocationI18n.setLocale(new Locale("en", "GB"));
  invocationI18n.setTimeZone(SimpleTimeZone.getTimeZone("GMT"));
}
catch (IllegalStateException ise) {
  // Is the component CMI; is the service started and enabled?
}
myEjb.myBusinessMethod();
```

Within CMI application components, the Internationalization and InvocationInternationalization interfaces are semantically equivalent. You can use either of these interfaces to obtain the context associated with the thread on which that component is running. For instance, both interfaces can be used to obtain the list of locales propagated to the servlet doPost service method.

Internationalization context API: Programming reference

Application components programmatically manage internationalization context through the UserInternationalization, Internationalization, and InvocationInternationalization interfaces in the com.ibm.websphere.i18n.context package.

The following code example introduces the internationalization context API:

```
public interface UserInternationalization {
  public Internationalization getCallerInternationalization();
  public InvocationInternationalization
  getInvocationInternationalization();
}

public interface Internationalization {
  public java.util.Locale[] getLocales();
  public java.util.Locale getLocale();
  public java.util.TimeZone getTimeZone();
}

public interface InvocationInternationalization
    extends Internationalization {
  public void setLocales(java.util.Locale[] locales);
  public void setLocale(java.util.Locale jmLocale);
  public void setTimeZone(java.util.TimeZonetimeZone);
  public void setTimeZone(String timeZoneId);
}
```

UserInternationalization interface

The UserInternationalization interface provides factory methods for obtaining references to the CallerInternationalization and InvocationInternationalization context objects. Use these references to access elements of the caller and invocation contexts correlated to the current thread.

Methods of the UserInternationalization interface:

Internationalization getCallerInternationalization()

Returns a reference implementing the Internationalization interface that supports access to

elements of the caller internationalization context correlated to the current thread. If the service is disabled, this method issues an IllegalStateException exception.

InvocationInternationalization getInvocationInternationalization()
Returns a reference implementing the InvocationInternationalization interface. If the service is disabled, this method issues an IllegalStateException exception.

Internationalization interface

The Internationalization interface declares methods that provide read-only access to internationalization context. Given a caller or invocation internationalization context object created with the UserInternationalization interface, bind the object to the Internationalization interface to get elements of that context type. Observe that caller internationalization context can be accessed only through this interface.

Methods of the Internationalization interface:

Locale[] getLocales()
Returns the chain of locales within the internationalization context (object) that is bound to the interface, provided the chain is not null; otherwise this method returns a chain of length(1) containing the default locale of the Java virtual machine (JVM).

Locale getLocale()
Returns the first in the chain of locales within the internationalization context (object) that is bound to the interface, provided the chain is not null; otherwise this method returns the default locale of the JVM.

TimeZone getTimeZone()
Returns the caller time zone (that is, the SimpleTimeZone instance) that is associated with the current thread, provided the time zone is non-null; otherwise this method returns the process time zone.

InvocationInternationalization interface

The InvocationInternationalization interface declares methods that provide read and write access to InvocationInternationalization context. Given an invocation internationalization context object created with the UserInternationalization interface, bind the object to the InvocationInternationalization interface to get and set elements of the invocation context.

According to the container-managed internationalization (CMI) policy, all set methods, setXxx(), issue an IllegalStateException exception when called within a CMI servlet or enterprise bean.

Methods of the InvocationInternationalization interface:

void setLocales(java.util.Locale[] locales)
Sets the chain of locales to the supplied chain, *locales*, within the invocation internationalization context. The supplied chain can be null or have length(>= 0). When the supplied chain is null or has length(0), the service sets the chain of invocation locales to an array of length(1) containing the default locale of the JVM. Null entries can exist within the supplied locale list, for which the service substitutes the default locale of the JVM on remote invocations.

void setLocale(java.util.Locale locale)
Sets the chain of locales within the invocation internationalization context to an array of length(1) containing the supplied locale, *locale*. The supplied locale can be null, in which case the service instead sets the chain to an array of length(1) containing the default locale of the JVM.

void setTimeZone(java.util.TimeZone timeZone)
Sets the time zone within the invocation internationalization context to the supplied time zone, *time zone*. If the supplied time zone is not an exact instance of java.util.SimpleTimeZone or is null, the service sets the invocation time zone to the default time zone of the JVM instead.

void setTimeZone(String timeZoneId)
Sets the time zone within the invocation internationalization context to the java.util.SimpleTimeZone having the supplied ID, *timeZoneId*. If the supplied time zone ID is null or

invalid (that is, the ID is not displayed in the list of IDs returned by the java.util.TimeZone.getAvailableIds method) the service sets the invocation time zone to the simple time zone having an ID of GMT, an offset of 00:00, and otherwise invalid fields.

Internationalization context:

An *internationalization context* is a distributable collection of internationalization information containing an ordered list, or chain, of locales and a single time zone, where the locales and time zone are instances of the java.util.Locale and java.util.TimeZone Java SDK types, respectively. A locale chain is ordered according to the user's preference.

The internationalization service manages and makes available two varieties of internationalization context: the *caller context*, which represents the caller's localization environment, and the *invocation context*, which represents the localization environment under which a business method runs. Server application components use elements of the caller and invocation internationalization contexts to appropriately tailor locale-sensitive and time zone-sensitive computations.

The internationalization service does not support time zone types other than the java.util.SimpleTimeZone type that is found in the Java SDK. Unsupported time zone types silently map to the default time zone of the JVM when supplied to internationalization context API methods. For a complete description of the java.util.Locale, java.util.TimeZone and java.util.SimpleTimeZone types, refer the Java SDK API documentation.

Caller context

Caller internationalization context contains the locale chain and time zone received on incoming EJB business method and servlet service method invocations; it is the internationalization context propagated from the calling process. Use caller context elements within server application components to localize computations to the calling component. Caller context is read-only and can be accessed by all application components by using the Internationalization interface of the internationalization context API.

Caller context is computed in the following manner: On an EJB business method or servlet service method invocation, the internationalization service extracts the internationalization context from the incoming request and scopes this context to the method as the caller context. For any missing or null context element, the service inserts the corresponding default element of the JVM (for example, java.util.Locale.getDefault() or java.util.TimeZone.getDefault().) The service performs a similar insertion whenever missing or null Caller context elements are encountered on invocations of stateless session beans that are enabled for Web services.

Formally, caller context is the invocation context of the calling business method or application component.

Invocation context

Invocation internationalization context contains the locale chain and time zone under which EJB business methods and servlet service methods run. It is managed by either the hosting container or the application component, depending on the applicable internationalization policy. On outgoing business method requests, it is the context that propagates to the target process. Use invocation context elements to localize computations under the specified settings of the current application component.

Invocation context is computed in the following manner: On an incoming business method or servlet service method invocation, the internationalization service queries the associated context management policy. If the policy is container-managed internationalization (CMI), the container scopes the context designated by the policy to the invocation; otherwise the policy is application-managed internationalization (AMI), and the container scopes an empty context to the invocation that can be altered by the method implementation.

Application components can access invocation context elements through both the Internationalization and InvocationInternationalization interfaces of the internationalization context API. Invocation context elements can be set (overwritten) under the application-managed internationalization policy only.

On an outgoing business method request, the service obtains the currently scoped invocation context and attaches it to the request. This outgoing exported context becomes the caller context of the target invocation. When supplying invocation context elements, either for export on outgoing requests or through the API, the internationalization service always provides the most recent element set using the API; the service also supplies the corresponding default element of the JVM for any null invocation context element.

Because the internationalization context that is propagated over Web services (SOAP) requests contains a time zone ID rather than the entire state of a java.lang.SimpleTimeZone object, time zone information might be lost when a Web service-enabled client program or session bean becomes involved in remote business computation.

Internationalization context: Propagation and scope:

The scope of internationalization context is implicit. Every Enterprise JavaBeans (EJB) client application, servlet service method, and EJB business method call has two internationalization contexts under which it runs.

For each application component call, the container enters the caller context and the call context, as indicated by the pertinent internationalization policy, into scope before the container delegates to the actual implementation. When the implementation returns, the service removes these contexts from scope. The internationalization service supplies no programmatic mechanism for components to explicitly manage the scope of internationalization context.

The service scopes internationalization context differently with respect to application component type:
- "EJB client programs (contained)"
- "Servlets" on page 487
- "Enterprise beans" on page 487
- "Web service client programs (contained)" on page 487
- "Stateless session beans that are enabled for Web services" on page 488

Internationalization context observes by-value semantics over remote method requests. Changes to internationalization context elements that are scoped to a call do not affect the corresponding elements of the internationalization context that is scoped to the remote calling process. Also, modifications to context elements obtained using the internationalization context API do not affect the corresponding elements that are scoped to the invocation.

EJB client programs (contained)

Before it calls the main method of a client program, the Java EE client container introduces into scope invocation and caller internationalization some contexts that contain null elements. These contexts remain in scope throughout the life of the program. EJB client programs are the base in a chain of remote business method invocations and, technically, do not have a logical caller context. Accessing a caller context element yields the corresponding default element of the client JVM. On outgoing EJB business method requests, the internationalization service propagates the invocation context to the target process. Any unset (null) invocation context elements are replaced with the default of the JVM when exported by the internationalization context API or by outgoing requests.

Tip:

To propagate values other than the JVM defaults to remote business methods, EJB client programs, as well as AMI servlets or enterprise beans, must set (override) elements of the invocation context. To learn how to set invocation context elements, see “Accessing invocation locales and time zones” on page 481.

Servlets

On every servlet service method (doGet or doPost) invocation, the Java EE Web container introduces caller and invocation internationalization contexts into scope before delegating to the service method implementation. The caller context contains the accept-languages propagated in the HTTP servlet request, typically from a Web browser. The invocation context contains whichever context is indicated by the container internationalization attribute of the internationalization policy that is associated with the servlet. Any unset (null) invocation context elements are replaced with the default of the server JVM when exported by the internationalization context API or by outgoing requests. The caller and invocation contexts remain effective until immediately after the implementation returns, at which time the container removes them from scope.

Enterprise beans

On every EJB business method invocation, the Java EE EJB container introduces caller and invocation internationalization contexts into scope before delegating to the business method implementation. The caller context contains the internationalization context elements imported from the incoming IIOP request; if the incoming request lacks a particular internationalization context element, the container scopes a null element. The invocation context contains whichever context is indicated by the container internationalization attribute of the internationalization policy that is associated with the business method.

On outgoing EJB business method requests, the service propagates the invocation context to the target process. Any unset (null) invocation context elements are replaced with the default of the server JVM when exported by the internationalization context API or by outgoing requests. The caller and invocation contexts remain effective until immediately after the implementation returns, when the container removes them from scope.

Consider a simple EJB application with a Java client that calls the remote myBeanMethod bean method. On the client side, the application can use the Internationalization Service API to set invocation context elements. When the client calls myBeanMethod(), the service exports the client invocation context to the outgoing request. On the server side, the service detaches the imported context from the incoming request and scopes it to the method as its caller context; the service also scopes the invocation context to the method as indicated by the associated internationalization context management policy. The EJB container then calls the myBeanMethod method, which can use the internationalization context API to access elements of either the caller or invocation contexts. When the myBeanMethod method returns, the EJB container removes these contexts from scope.

Web service client programs (contained)

Before it calls the main method of a Web service client program, the client container introduces into scope both invocation and caller internationalization contexts that contain null elements. These contexts remain in scope throughout the duration of the program. Web service client programs are the base in a chain of remote business method invocations and, technically, do not have a logical caller context. Accessing a Caller context element yields the corresponding default element of the client virtual machine.

On outgoing Web service requests, the internationalization service transparently creates a SOAP header block that contains the invocation context that is associated with the current thread; the SOAP representation of invocation context is propagated through the request to the target process. Any unset (that is, null) invocation context elements are replaced with the default element of the JVM when exported by the internationalization context API or by outgoing requests. Also, because the header contains only a

time zone ID, the additional state of the time zone object (java.lang.SimpleTimeZone) of the invocation context might be lost, because it does not get propagated through the request.

Tip:

To propagate values other than the JVM defaults to remote business methods, Web service client programs, as well as AMI servlets or enterprise beans, must set (override) elements of the invocation context. For more information, see “Accessing invocation locales and time zones” on page 481.

Stateless session beans that are enabled for Web services

On every method invocation of a Web service-enabled bean, the EJB container introduces caller and invocation internationalization contexts into scope before delegating control to the business method implementation. The caller context contains the internationalization context elements that are imported from the SOAP header block of the incoming request. If the incoming request lacks a particular internationalization context element, the container introduces a null element into scope. The invocation context contains whichever context is indicated by the container internationalization attribute of the internationalization policy that is associated with the business method.

On outgoing EJB business method requests, the service propagates the invocation context to the target process. Any unset (that is, null) invocation context elements are replaced with the default element of the server JVM when exported by the internationalization context API or by outgoing requests. The caller and invocation contexts remain effective until immediately after control returns from the business method implementation, at which time the container removes them from scope.

On outgoing Web service requests, the internationalization service transparently creates a SOAP header block that contains the invocation context associated with the current thread. The SOAP representation of the invocation context is propagated through the request to the target process. Any unset (that is, null) invocation context elements are replaced with the default element of the JVM when exported by the internationalization context API or by outgoing requests.

Thread association considerations

The Web and EJB containers scope internationalization contexts to a method by associating the method with the thread that run the method implementation. Similarly, methods of the internationalization context API either associate context with, or obtain context associated with, the thread on which these methods run.

In cases where new threads are spawned within an application component (for instance, a user-generated thread inside the service method of a servlet, or a system-generated event handling thread in an AWT client) the internationalization contexts associated with the parent thread does not automatically transfer to the newly-spawned thread. In such instances, the service exports the default locale and time zone of the JVM on any remote business method request and on any API calls that run on the new thread.

If the default context is inappropriate, the desired invocation context elements must be explicitly associated to the new thread by using the setXxx methods of the InvocationInternationalization interface. Currently, internationalization context management policies enable invocation context to be set within EJB client programs, as well as within servlets, session beans, and message-driven beans that use application-managed internationalization.

Internationalization context: Management policies:

Internationalization policies prescribe how Java EE application components or their hosting containers manage internationalization context on component invocations. Two internationalization context management policies apply to all component types: Application-managed internationalization (AMI) and Container-managed internationalization (CMI).

These policies are represented in two parts:
- Internationalization type
- Container internationalization attribute

The service defines a default, or implicit, internationalization policy for every application component type. At development time, assemblers can override the default policy for server component types by explicitly configuring their internationalization type, and optional container internationalization attributes. Policies configured during assembly are preserved in the deployment descriptor for the application.

All components have an internationalization type that indicates whether it is AMI or CMI; that is, whether a component is to deploy under the application-managed or the container-managed internationalization policy. Application assemblers can set the internationalization type for servlets, session beans, and message-driven beans. Entity beans are implicitly CMI and EJB clients are implicitly AMI; neither can be configured otherwise.

For CMI servlets and enterprise beans, optional container internationalization attributes can be specified to indicate which invocation internationalization context the container is to scope to service or business methods. A CMI service or business method invocation can run under the context of the caller's process, under the default context of the server JVM, or under a custom context specified in the attribute. Assemblers can specify one container internationalization attribute per disjoint set of CMI servlets within a Web module, or one Attribute per disjoint set of business methods of CMI beans within an EJB module. A container internationalization attribute can be associated with more than one method, but a method cannot be associated with more than one attribute.

When an application server launches an application, the internationalization service collects policy information from the deployment descriptor, then uses this information to construct and associate an internationalization policy to every component invocation. A policy is denoted as:

```
[<Internationalization Type>,<Container Internationalization Attribute>]
```

Several cases exist in which the deployment descriptor seems to lack policy information, for example: EJB client applications have no configurable internationalization policy settings; AMI components do not have container internationalization attributes; and you are not required to specify container internationalization attributes for CMI components. When the service cannot obtain the explicit internationalization type and container attribute settings from a well-formed deployment descriptor, it implicitly inserts the appropriate setting into the policy.

The service observes the following conventions when applying policies to invocations:
- Servlets (service) and EJB business methods lacking all internationalization policy information in the deployment descriptor implicitly run under policy `[CMI,RunAsCaller]`.
- CMI servlets and business methods lacking a container internationalization attribute in the deployment descriptor implicitly run under policy `[CMI,RunAsCaller]`.
- AMI servlets and business methods always lack container internationalization attributes in the deployment descriptor, but implicitly run under the logical policy `[AMI,RunAsServer]`.
- EJB clients always lack internationalization policy information in the deployment descriptor. By definition, EJB clients are implicitly AMI types and run under the invocation context of the JVM; they run under the logical policy `[AMI,RunAsServer]`.

For conditions other than these cited examples, such as a malformed deployment descriptor, refer to Internationalization service errors.

Internationalization policies for EJB clients and HTTP clients cannot be configured; HTTP clients do, however, run under the language priority settings of the hosting Web browser. These settings are configurable under the options dialog of most Web browsers. Refer to your Web browser documentation for details.

Internationalization type:

Every server application component has an *internationalization type* setting that indicates whether the invocation internationalization context is managed by the component or by the hosting Java EE container.

Server application components can be deployed to use one of two types of internationalization context management:
- Application-managed internationalization (AMI)
- Container-managed internationalization (CMI)

A server component can be deployed as AMI or CMI, but not both; CMI is the default. The setting applies to the entire component on every invocation. Entity beans use CMI only. Enterprise JavaBeans (EJB) client applications do not have an internationalization type setting; they implicitly use AMI.

Application-managed internationalization

Under the AMI deployment policy, component developers assume complete control over the invocation internationalization context. AMI components can use the internationalization context API to programmatically set invocation context elements.

AMI components are expected to manage invocation context. Invocations of AMI components implicitly run under the default locale and time zone of the hosting JVM. Invocation context elements not set using the API default to the corresponding elements of the JVM when accessed through the API or when exported on business methods. To export context elements other than the JVM defaults, AMI servlets, AMI enterprise beans, and EJB client applications must set (overwrite) invocation elements using the internationalization context API. Moreover, the container logically suspends the caller context that is imported on the AMI servlet lifecycle method and AMI EJB business method invocations. To continue propagating the context of the calling process, AMI servlets and enterprise beans must use the API to transfer caller context elements to the invocation context.

Specify AMI for server components that have internationalization context management requirements that are not supported by container-managed internationalization (CMI).

Container-managed internationalization

CMI is the preferred internationalization context management policy for server application components; it is also the default policy. Under CMI, the internationalization service collaborates with the Web and EJB containers to set the invocation internationalization context for servlets and enterprise beans. The service sets invocation context according to the container internationalization attribute of the policy that is associated with a servlet (service method) or an EJB business method.

A CMI policy has a container internationalization attribute that indicates which internationalization context the container is to scope to an invocation. For details, see Container internationalization attributes. By default, invocations of CMI components run under the caller's internationalization context; or rather, they adhere to the implicit policy `[CMI,RunasCaller]` whenever the servlet or business is not associated with an attribute in the deployment descriptor. For complete details, see Internationalization context: Management policies.

Methods within CMI components can obtain elements of the invocation context using the internationalization context API, but cannot set them. Any attempt to set invocation context elements within CMI components results in a java.lang.IllegalStateException exception.

Specify container-managed internationalization for server application components that require standard internationalization context management. Then specify the container internationalization attributes for CMI servlets and for business methods of CMI enterprise beans that you do not want to run under the caller's internationalization context.

Container internationalization attributes:

The internationalization policy of every CMI servlet and EJB business method has a *container internationalization attribute* that specifies which internationalization context the container is to scope to its invocation.

The container internationalization attribute has three main fields:
- Run as
- Locales
- Time zone ID

As a convenience, you can create named container internationalization attributes and associate them to the following subsets:
- CMI servlets within a Web module
- Business methods of CMI enterprise beans within an Enterprise JavaBeans (EJB) module
- Business methods of Web service-enabled session beans. In the following descriptions, the term *supported enterprise bean* refers to both CMI enterprise beans and Web service-enabled session beans.

Run-as field

The **Run-as** field specifies one of three types of invocation context that a container can scope to a method. For servlet service and EJB business methods, the container constructs the invocation internationalization context according to the **Run as** field setting and associates this context to the current thread before delegating to the method implementation.

By default, invocations of servlet service methods and EJB business methods implicitly run as caller (`RunAsCaller`) unless the **Run as** field of a policy attribute specifies otherwise. EJB client applications and AMI server components always run as server (`RunAsServer`).

You can specify the following invocation context types with the **Run as** field are:

Caller The container calls the method under the internationalization context of the calling process. For any missing context element, the container supplies the corresponding default context element of the Java virtual machine (JVM). Select run as caller when you want the invocation to run under the invocation context of the calling process.

Server
The container calls the method under the default locale and time zone of the JVM. Select run as server when you want the invocation to run under the invocation context of the JVM.

Specified
The container calls the method under the internationalization context specified in the attribute. Select run as specified when you want the invocation to run under the custom invocation context that is specified in the policy; then provide the custom context elements by completing the Locales and Time zone ID fields.

Remember: Java Message Service (JMS) messages do not contain internationalization context. Although container-managed message-driven beans can be configured to run as caller, the container associates the default elements of the server process when calling the onMessage method of any message-driven bean that is configured as `[CMI, RunAsCaller]`. You can also configure the **Run as** field for Web service business methods.

Locales field

The **Locales** field specifies an ordered list of locales that the container scopes to an invocation. A locale represents a specific geographical, cultural, or political region and contains three fields:
- **Language code**. Ideally, language code is one of the lower-case, two-character codes that are defined by the ISO 639 standard; however, language code is not restricted to ISO codes and is not a required field. A valid locale must specify a language code if it does not specify a country code.

- **Country code**. Ideally, country code is one of the upper-case, two-character codes that are defined by the ISO 3166 standard; however, country code is not restricted to ISO codes and is not a required field. A valid locale must specify a country code if it does not specify a language code.
- **Variant**. Variant is a vendor-specific code. Variant is not a required field and serves only to supplement the language and country code fields according to application- or platform-specific requirements.

A valid locale must specify at least a language code or a country code; the variant is always optional. The first locale of the list is returned when accessing invocation context using the getLocale method of the internationalization context API.

Time zone ID field

The **Time zone ID** field specifies an abbreviated identifier for a time zone that the container scopes to an invocation. You can also configure the **Time zone ID** field for Web service business methods.

A time zone represents a temporal offset and computes daylight savings information. A valid ID indicates any time zone supported by the java.util.TimeZone type. Specifically, a valid ID is any of the IDs that appear in the list of time zone IDs returned by method java.util.TimeZone.getAvailableIds(), or a custom ID having the form `GMT[+|-]hh[[:]mm]`; for example, `America/Los_Angeles`, `GMT-08:00` are valid time zone IDs.

Administering the internationalization service

To use internationalization context in an Enterprise JavaBeans (EJB) application, the internationalization service must be enabled in the runtime environments for all server-side components (servlets and enterprise beans, including session beans enabled for Web service usage) as well as all client-side components (EJB client applications and Web service clients).

About this task

If you do not require the internationalization service, do not enable it. Leaving the service disabled prevents any possible performance degradation incurred by the implicit distribution of internationalization resources.

The internationalization service cannot be enabled for HTTP clients, because support for internationalization in that case is provided by the browser, not by the application server.

Procedure

- Enable or disable the internationalization service for servlets and enterprise beans. By default, the service is disabled for server-side components within the application server. You enable the service by using either the administrative console or the wsadmin tool.
- Enable or disable the internationalization service for EJB clients. By default, the service is disabled within the client container. You enable the service by using the launchClient tool.

Enabling the internationalization service for servlets and enterprise beans

Perform this task to enable the internationalization service in the application server runtime environment.

About this task

Any servlet or enterprise bean can use internationalization context if the internationalization service is enabled within the hosting application server instance.

Procedure

1. Start the administrative console.
2. Click **Servers** > **Application servers** > ***server_name*** > **Container services** > **Internationalization service**.

3. Enable the internationalization service.
   a. If not already selected, select the **Enable service at server startup** check box.
   b. Click **OK**.

Results

When you select the **Enable service at server startup** setting, the application server automatically initializes and starts the internationalization service whenever the server starts. If you change this setting, be sure to restart the application server for the new setting to take effect.

To disable the service, clear the **Enable service at server startup** check box. In this case, the internationalization service is initialized but not started when the application server starts.

Example

Alternatively, the internationalization service can be enabled from the command line by using the wsadmin tool. Start the wsadmin tool and enter the following commands:

```
set x [$AdminConfig list I18NService]
$AdminConfig modify $x { { enable true } }
$AdminConfig save
exit
```

What to do next

If you enable or disable the internationalization service, be sure to stop and then restart the application server for the new setting to take effect.

Enabling the internationalization service for EJB clients

By default, the internationalization service is disabled for use within Enterprise JavaBeans (EJB) and Web-service enabled client applications. You must enable the service for client applications as well as for all server instances in the runtime environment.

Procedure

Enable the service.
When calling the launchClient tool, include the argument `-CCDI18NService.enable=true` or `-CCDI18NService.enable=yes`.

Internationalization service settings

Use this page to enable or disable the internationalization service. The internationalization service manages the implicit propagation and scoping of locale and time zone information, called *internationalization context*, within application components. When the service is enabled, application components can use the internationalization context API to programmatically manage locale and time zone information. In turn, components can use that locale and time zone information with the Java Platform, Standard Edition (JSE) Internationalization API to perform localizations. If internationalization support is not required on the server, disabling the service can improve performance.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers >** *server_name*. Then, under Container Settings, click **Container Services > Internationalization Service**.

Enable service at server startup:

Specifies whether the server attempts to start the internationalization service.

Default Cleared

Range Valid values are Selected or Cleared

More information about valid values follows:

Selected
When the application server starts, it attempts to start the internationalization service automatically.

Cleared
The server does not try to start the internationalization service.

To enable the internationalization service for applications on this server, the system administrator must select this property and then restart the server.

Internationalization service errors

Certain conditions might cause the internationalization service not to start, to issue java.lang.IllegalStateException exceptions while an application is running, or to exercise default behaviors.

The java.lang.IllegalStateException exception indicates one of the following things:

- An application component attempted an operation that is not supported by the internationalization programming model.

  The IllegalStateException exception is issued whenever a server application component whose internationalization type is set to container-managed internationalization (CMI) attempts to set invocation context. This behavior is a violation of the CMI policy, under which servlets and enterprise beans cannot modify their invocation internationalization context.
- An anomaly occurred that disabled the service.

  For instance, if the internationalization service is not properly initialized, the Java Naming and Directory Interface (JNDI) lookup on the UserInternationalization URL attribute issues a javax.naming.NameNotFoundException exception that contains an IllegalStateException instance.

The following conditions can occur while your internationalized application is running. These conditions might cause the internationalization service not to start, to issue IllegalStateException exceptions, or to exercise default behaviors:

- "The service is disabled "
- "The service is not started" on page 495
- "Invalid context element" on page 496
- "Missing context element" on page 496
- "Invalid policy" on page 496
- "Missing policy" on page 496

If you encounter unexpected or exceptional behavior, the problem is likely related to one of these conditions. You need to examine the trace log to investigate these conditions, which requires that you configure the diagnostic trace service to generate messages about internationalization service function.

The trace strings for the internationalization service follow; use both:

```
com.ibm.ws.i18n.context.*=all=enabled:com.ibm.websphere.i18n.context.*=all=enabled
```

The service is disabled

The internationalization service is not initialized when the startup setting is cleared. The service generates a message that indicates whether it is enabled or disabled. Applications cannot access the internationalization API when the service is disabled. If an application attempts a JNDI lookup to obtain the UserInternationalization reference, the lookup fails with a NamingException exception, indicating that the reference cannot be found. In addition, the service does not scope (propagate) internationalization context on incoming (outgoing) business method calls.

The service is not started

The internationalization service is operational whenever it is in the STARTED state. For example, if an application attempts to access internationalization context and the service is not started, the API issues an IllegalStateException exception. In addition, the service does not provide runtime support for servlets and enterprise beans.

As an application server progresses through its life cycle, it initializes, starts, stops, and terminates (destroys) the internationalization service. If an anomaly occurs during initialization, the service does not start. After the service is started, its state can change to BLOCKED in the event that a serious error occurs. The service generates a message for every state change.

If a trace message indicates that the service is not STARTED, examine previous messages to determine the problem. For instance, the internationalization service does not start if the activity service is unavailable and a message is displayed to that effect during initialization of the internationalization service.

During startup, the following messages indicate potential configuration or runtime problems:

No ORB support
: The service cannot obtain an instance of the object request broker (ORB). This condition is a fatal error. Examine the `SystemErr.log` and `SystemOut.log` files for information.

No TCM support
: The service cannot obtain an instance of its thread context manager (TCM). This condition is a fatal error. Examine the `SystemErr.log` and `SystemOut.log` files for information.

No IIOP (activity service) support
: The service cannot register with the activity service. This condition is a fatal error. The internationalization service cannot propagate or receive context on Internet Inter-ORB Protocol (IIOP) requests without activity service support. Examine the `SystemErr.log` and `SystemOut.log` files for information.

No AsynchBeans support
: The service cannot register into the asynchronous beans environment. This warning indicates that the asynchronous beans environment cannot support internationalization context.

No EJB container support
: The service cannot register with the Enterprise JavaBeans (EJB) container. This warning indicates that the internationalization service cannot support enterprise beans. Without EJB container support, internationalization contexts do not scope properly to EJB business methods. Review the trace log for any EJB container-related error conditions.

No Web container support
: The service cannot register with the Web container. This warning indicates that the internationalization service cannot support servlets and JavaServer Page (JSP) files. Without Web container support, internationalization contexts do not scope properly to servlet service methods. Review the trace log for any Web container-related error conditions.

No Metadata support
: The service cannot register with the metadata service. This warning indicates that the internationalization service cannot process the internationalization policies within application deployment descriptors. Without metadata support, the service associates the default internationalization context management policy, [CMI, RunAsCaller], to every servlet lifecycle method and enterprise bean business method invocation. Review the trace log for any metadata service-related error conditions.

No JNDI (Naming service) support
: The service cannot bind the UserInternationalization object into the namespace. This condition is a fatal error. Application components are unable to access internationalization context API references, and are therefore unable to access internationalization context elements. Review the trace log for any Naming (JNDI) service-related error conditions.

No API support
: The service cannot obtain an instance of an internationalization context API object. This condition

is a fatal error. Application components are unable to access internationalization context API references, and are therefore unable to access internationalization context elements.

Invalid context element

The service detected an invalid internationalization context element. For example, the internationalization service does not support TimeZone instances of a type other than java.util.SimpleTimeZone. If the service encounters an unusable element, it logs a message and substitutes the corresponding default element of the JVM.

Missing context element

The service detected a missing internationalization context element. Incoming requests (for example, from application servers that do not support the internationalization service) lack internationalization context. When the service attempts to access a caller internationalization context element (which does not exist in this case), the service logs a message and substitutes the corresponding default element of the Java virtual machine (JVM).

Whenever possible, enable the internationalization service within all clients and hosting application servers that comprise an internationalized enterprise application. Read more information about Administering the internationalization service in the *Administering applications and their environment* PDF book.

Invalid policy

The internationalization service detected a malformed internationalization policy in the application deployment descriptor. The service replaces the malformed attribute with the appropriate default. For instance, if the internationalization type for an entity bean is set to `Application` during the run of a servlet or EJB business method call, the service logs the inconsistency and enforces the `Container` setting instead.

Also, AMI application components do have an implicit container internationalization attribute. By default they run as server. The service silently enforces the implicit policy, `[AMI, RunAsServer]`, and logs messages to this effect.

Invalid container internationalization attributes are likely to occur when specifying the Locales and Time zone ID fields. When encountering invalid locales and time zone IDs within attributes, the service replaces each value with the corresponding default element of the JVM. Be sure to follow the guidelines provided in the *Developing and deploying applications* PDF book.

Missing policy

The service detected a missing internationalization policy. The service replaces the missing policy with the appropriate default. For instance, if the internationalization type is missing for a servlet or enterprise bean, the service sets the attribute to Container.

Container internationalization attributes are not mandatory for CMI application components. In the event that a CMI servlet or EJB business method lacks a container internationalization attribute, the service silently enforces the implicit policy `[CMI, RunAsCaller]`.

When an application lacks internationalization policies in its deployment descriptor, or metadata support is unavailable, the service logs a message and applies the policy `[CMI, RunAsCaller]` on every servlet service method and EJB business method invocation.

Read the information in the *Developing and deploying applications* PDF book:
- Assembling internationalized applications
- Container internationalization attributes

- Internationalization type

Chapter 11. Administering Mail, URLs, and other Java EE resources

This page provides a starting point for finding information about resources that are used by applications that are deployed on a Java Enterprise Edition (Java EE)-compliant application server. They include:

- JavaMail support for applications to send Internet mail
- URLs, for describing logical locations
- Resource environment entries, for mapping logical names to physical names
- Java DataBase Connectivity (JDBC) resources and other technology for data access (discussed elsewhere)
- Java Message Service (JMS) resources and other messaging system support (discussed elsewhere)

Configuring mail providers and sessions

Configure your own mail providers and sessions to customize how mail is handled in the application server. A mail provider encapsulates a collection of protocol providers, like SMTP, IMAP and POP3, and others. Mail sessions authenticate users and control access to messaging systems.

About this task

The application server includes a default mail provider that is called the built-in provider. If you use the default mail provider, you only have to configure the mail session.

To use a customized mail provider, you must create the mail session and provider.

Procedure

- Create the mail session.
  1. In the administrative console, click **Resources** > **Mail** > **Mail sessions**.
  2. Select the scope for the new mail session.
  3. Click **New**.
  4. Type the mail session name in the Name field.
  5. Type the JNDI name in the JNDI Name field.
  6. Optional: Enable strict Internet address parsing. This option specifies whether the recipient addresses must be parsed in strict compliance with RFC 822, which is a specifications document that is issued by the Internet Architecture Board. This setting is not generally used for most mail applications, and by default this setting is not enabled.

     RFC 822 syntax for parsing addresses effectively enforces a strict definition of a valid email address. If you select this setting, your mail component adheres to RFC 822 syntax and rejects recipient addresses that do not parse into valid email addresses as defined by the specification. If you do not select this setting, your mail component does not adhere to RFC 822 syntax and accepts recipient addresses that do not comply with the specification. You can view the RFC 822 specification at the website for the World Wide Web Consortium.
  7. Optional: Enable debug mode. Select this option to print interaction between the mail application and the mail servers and the properties of this mail session to the SystemOut.log file.

     Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer

command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

8. Provide information for the incoming mail service, outgoing service, or both. Enter the following information in the fields that are provided:
   - Server
   - Protocol
   - User
   - Password
   - Return email address. This field is available for outgoing mail properties.
9. Click **Apply** or **OK**.

- Create the mail provider, and optionally define one or more protocol providers. In the administrative console, click **Resources** > **Mail** > **Mail Providers**.
  1. Select the scope for the new mail provider.
  2. Click **New**.
  3. Type the name of the mail provider in the name field.
  4. Optional: Isolate the mail provider.

     You can isolate a mail provider to allow different versions of the same provider to be loaded in the same Java Virtual Machine (JVM). For example, you might want to deploy multiple applications on a single server, but each application requires different versions or implementations of the mail provider. You can isolate each version or implementation of the provider, and the provider is loaded in its own class loader and does not interfere with other implementations. There are some general considerations for isolating any type of resource provider; refer to the topic on considerations for isolated resource providers for more information.
     a. Select **Isolate this mail provider**.
     b. Give the mail provider a unique class path that is appropriate for that version or implementation.
  5. Click **Apply** or **OK**.
  6. Define one or more protocol providers for the mail provider.
     a. Click *mail_provider*.
     b. Click **Protocol Providers**.
     c. Click **New**.
     d. Type the protocol name in the **Protocol** field.
     e. Type the class name in the **Class name** field.
     f. Select the type of mail server that this protocol provider supports. Select TRANSPORT or STORE. TRANSPORT corresponds to outgoing mail services, and STORE corresponds to incoming mail services.
     g. Click **Apply** or **OK**.

     Ensure that every mail session is defined under a parent mail provider. Select a mail provider first and then create your mail session.
- Optional: Configure the mail session.
  1. Click *mail_provider*.
  2. Click **Mail Sessions**.
  3. Click *mail_session*.
  4. Make changes to appropriate fields.
  5. Click **Apply** or **OK**.

What to do next

If your application has a client, you can configure mail providers and sessions using the Application Client Resource Configuration Tool.

Mail provider collection

Use this page to view available JavaMail service providers, also known as *mail providers*. The mail provider encapsulates a collection of protocol providers, which implement the protocols for communication between your mail application and mail servers.

To view this administrative console page, click **Resources** > **Mail** > **Mail Providers**.

The **built-in mail provider** made available by WebSphere Application Server encompasses three protocol providers: Simple Mail Transfer Protocol (SMTP), Internet Message Access Protocol (IMAP) and Post Office Protocol (POP3). Select the built-in provider if these protocols provide the right support for your mail system. If you have installed or plan to install different protocol providers, you must assign them a mail provider; select the scope at which you want the new mail provider to implement the protocols, then select **New**.

Name
Specifies the name of the JavaMail resource provider.

Scope
Specifies the scope in which the mail provider supports installed mail applications.

Description
Specifies the resource provider description.

Mail provider settings

Use this page to edit mail provider properties or configure a new mail provider. The mail provider encapsulates a collection of protocol providers, which implement the protocols for communication between your mail application and mail servers.

To view this administrative console page, click **Resources** > **Mail** > **Mail Providers** > ***mail_provider***, or create a new mail provider by clicking **Resources** > **Mail** > **Mail Providers** > **New**.

Scope
Specifies the scope in which the mail provider supports installed mail applications.

Name
Specifies the name of the mail provider.

Description
Specifies the resource provider description.

Isolate this mail provider
Specifies that this mail provider will be loaded in its own class loader. This allows the application server to load different versions or implementations of the same mail provider in the same Java Virtual Machine. Give each version or implementation of the mail provider a unique class path that is appropriate for that version or implementation.

Class path
Specifies the class path to a Java archive (JAR) file that contains the implementation classes for this mail provider. If more than one JAR file provides the complete implementation, add an entry for each JAR file that the mail provider requires. Enter one class path per line; do not use class path separator information.

Protocol providers collection

Use this page to select or add a protocol provider that supports interaction between your mail application and mail servers. For example, your application might require the Simple Mail Transfer Protocol (SMTP), which is a popular transport protocol for sending mail. Selecting that protocol provider allows your mail application to connect and send mail through the server.

To view this administrative console page, click **Resources** > **Mail** > **Mail Providers** > ***mail_provider*** > **Protocol Providers**.

Protocol

Specifies the configuration of the protocol provider for a given protocol.

Class name

Specifies the implementation class for the specific protocol provider, which is also known as the mail service provider.

Type

Specifies the type of protocol provider. Valid options are **STORE** or **TRANSPORT**.

Protocol providers settings

Use this page to set properties of a protocol provider, including SMTP, IMAP and POP3, which provides the implementation class for a specific protocol to support communication between your mail application and mail servers.

Built-in providers: The application server contains protocol providers for various types of protocols, including SMTP, IMAP and POP3. If you require custom providers for protocols that are not provided by the application server, install them in your application serving environment before configuring the providers. See the JavaMail API design specification for guidelines. After configuring your protocol providers, return to the mail provider page to find the link for configuring mail sessions.

To view this administrative console page, click **Resources** > **Mail** > **Mail Providers** > ***mail_provider*** > **Protocol Providers** > ***protocol_provider***.

Scope

Specifies the scope at which this protocol provider was created. Only applications that are installed within this scope can use a protocol that is configured according to the settings of this protocol provider.

Protocol

Specifies the configuration of the protocol provider for a given protocol.

Class name

Specifies the implementation class of this protocol provider.

Type

Specifies the type of protocol provider. Valid options are **STORE** or **TRANSPORT**.

Mail session collection

Use this page to view mail sessions that are defined under the parent mail provider.

You can access this administrative console page in one of two ways:

- **Resources** > **Mail** > **Mail Sessions**
- **Resources** > **Mail** > **Mail Providers** > *mail_provider* > **Mail Sessions**

Name
Specifies the administrative name of the JavaMail session object.

Scope
Specifies the scope at which the mail session was created. Only JavaMail applications that are installed in this scope can use this mail session.

Provider
Specifies the mail provider that WebSphere Application Server uses for this mail session.

JNDI Name
Specifies the Java Naming and Directory Interface (JNDI) name for the resource, including any naming subcontexts.

This name provides the link between the platform binding information for resources defined in the client application deployment descriptor and the actual resources bound into JNDI by the platform.

Description
Specifies an optional description for your administrative records.

Category
Specifies an optional collection for classifying or grouping sessions.

Mail session configuration settings
Use this page to configure mail sessions.

You can access this administrative console page in one of two ways:

- **Resources** > **Mail** > **Mail sessions** > ***mail_session***
- **Resources** > **Mail** > **Mail Providers** > ***mail_provider*** > **Mail sessions** > ***mail_session***

Scope
Specifies the scope of the mail provider that implements the JavaMail API for this mail session. Only applications that you installed within this scope can use this mail session.

Provider
Specifies the mail provider that the application server uses for this mail session.

When you create a mail session, if you previously defined one or more mail providers at the relevant scope, you will see a list from which you can select an existing mail provider for the new mail session.

Create New Provider
Provides the option of configuring a new mail provider for the new mail session.

Create New Provider is displayed only when you click **Resources** > **Mail** > **Mail sessions** > **New** to create a new mail session.

Clicking **Create New Provider** triggers the console to display the mail provider configuration page, where you create a new provider. After you click **OK** to save your settings, you see the mail session collection page. Click **New** to define a new mail session for use with the new provider; the console now displays a configuration page that lists the new mail provider as the mail session Provider.

Note: After you create a mail session, you cannot change the provider of that mail session.

Name
Specifies the administrative name of the JavaMail session object.

JNDI name
Specifies the Java Naming and Directory Interface (JNDI) name for the resource, including any naming subcontexts.

This name provides the link between the platform binding information for resources that are defined in the client application deployment descriptor and the actual resources bound into JNDI by the platform.

Important: Adhere to the following requirements for JNDI names:

- Do not assign duplicate JNDI names across different resource types (such as mail sessions versus data sources).
- Do not assign duplicate JNDI names for multiple resources of the same type in the same scope.

Description
Specifies an optional description for your administrative records.

Category
Specifies an optional collection for classifying or grouping sessions.

Enable debug mode
Toggles debug mode on and off for this mail session.

Enable strict Internet address parsing
Specifies whether the recipient addresses must be parsed in strict compliance with RFC 822, which is a specifications document issued by the Internet Architecture Board.

This setting is not generally used for most mail applications. RFC 822 syntax for parsing addresses effectively enforces a strict definition of a valid email address. If you select this setting, your mail component adheres to RFC 822 syntax and rejects recipient addresses that do not parse into valid email addresses (as defined by the specification). If you do not select this setting, the mail component does not adhere to RFC 822 syntax and accepts recipient addresses that do not comply with the specification. By default, this setting is not selected. You can view the RFC 822 specification at the World Wide Web Consortium website.

Outgoing Mail Properties

Server:

Specifies the server that is accessed when sending mail.

Protocol:

Specifies the protocol to use when sending mail. Actual protocol values are defined in the protocol providers that you configured for the current mail provider.

User:

Specifies the user of the mail account when the outgoing mail server requires authentication.

This setting is not generally used for most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.

Password:

Specifies the password to use when the outgoing mail server requires authentication.

This setting is not generally used for most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.

Verify Password:

Verify the password.

Return email address:

Specifies the Internet email address that is displayed in messages as the mail originator.

This value represents the Internet email address that, by default, displays in the received message in the **From** or the **Reply-To** address. The recipient's reply will come to this address.

Incoming Mail Properties

Server:

Specifies the server that is accessed when receiving mail.

Protocol:

Specifies the protocol to use when receiving mail. Actual protocol values are defined in the protocol providers that you configured for the current mail provider.

User:

Specifies the user of the mail account when the incoming mail server requires authentication.

This setting is not generally used for most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.

Password:

Specifies the password to use when the incoming mail server requires authentication.

This setting is not generally used for most mail servers. Leave this field blank unless you use a mail server that requires a user ID and password.

Verify Password:

Verify the password.

Administering URLs

URL provider collection

Use this page to view existing URL providers, which supply the implementation classes that are necessary for WebSphere Application Server to access a URL through a specific protocol. The default URL provider provides connectivity through protocols that are supported by the IBM Developer Kit for the Java™ Platform, compatible with the Java 2 Standard Edition Platform 1.3.1. These protocols include HyperText Transfer Protocol (HTTP) and File Transfer Protocol (FTP), which work for must URLs.

To view this administrative console page, click **Resources** > **URL** > **URL Providers**.

Name

Specifies the administrative name for the URL provider.

Scope
Specifies the scope of this URL provider, which can support multiple URL configurations. All of the URL configurations that are supported by this provider inherit this scope.

Description
Describes the URL provider for your administrative records.

URL provider settings
Use this page to configure URL providers, which support WebSphere Application Server connections to a URL over a specific protocol.

To view this administrative console page, click **Resources** > **URL** > **URL Providers** > *URL_provider*.

Scope
Specifies the scope of this URL provider, which can support multiple URL configurations. All of the URL configurations that are supported by this provider inherit this scope.

Name
Specifies the administrative name for the URL provider.

Description
Describes the URL provider, for your administrative records.

Class path
Specifies paths or JAR file names which together form the location for the resource provider classes.

Stream handler class name
Specifies fully qualified name of a user-defined Java class that extends the `java.net.URLStreamHandler` class for a particular URL protocol, such as FTP.

Protocol
Specifies the protocol supported by this stream handler. For example, NNTP, SMTP, FTP.

URL configurations collection
Use this page to view existing Uniform Resource Locator (URL) configurations, which are sets of properties that define WebSphere Application Server connections to URLs. URLs are location names that represent electronically accessible resources, such as a directory file on a machine in a network or a document stored in a database.

You can access this administrative console page in one of two ways:

- **Resources** > **URL Providers** > *URL_provider* > **URLs**
- **Resources** > **URL** > **URLs**

Name
Specifies the display name for the resource.

JNDI Name
Specifies the JNDI name.

Scope
Specifies the scope of the URL provider that supports this URL configuration. Only applications that are installed within this scope can use this URL configuration to access URL resources.

Provider
Specifies the URL provider that supplies the implementation classes for using a specific protocol to access this URL.

Description
Specifies the description of the resource.

Category
Specifies the category string, which you can use to classify or group the resource.

URL configuration settings
Use this page to define connections to Uniform Resource Locators (URLs), which are location names that represent electronically accessible resources. A collection of URL connection properties is often called a URL configuration in the WebSphere Application Server environment. The targeted resources are remote to your Application Server installation.

You can access this administrative console page in one of two ways:

- **Resources** > **URL** > **URLs** > *URL*
- **Resources** > **URL** > **URL Providers** > *URL_provider* > **URLs** > *URL*

Scope
Specifies the scope of the URL provider that supports this URL configuration. Only applications that are installed within this scope can use this URL configuration to access URL resources.

Provider
Specifies the URL provider that WebSphere Application Server uses for this URL configuration.

To create a new URL configuration: If you previously defined one or more URL providers at the relevant scope, you see a list from which you can select an existing URL provider for your new URL configuration.

Create New Provider
Provides the option of configuring a new URL provider for the new URL configuration.

Create New Provider is displayed only when you create a new URL from the **Resources** > **URL** > **URLs** path. In this flow, you can create a new URL provider if needed. The URL provider can not be changed during an edit.

Clicking **Create New Provider** triggers the console to display the URL provider configuration page, where you create a new provider. After you click **OK** to save your settings, you see the URL collection page. Click **New** to define a new URL configuration for use with the new provider; the console now displays a configuration page that lists the new provider as the URL configuration Provider.

Name
Specifies the display name for the resource.

JNDI Name
Specifies the JNDI name.

Important: Adhere to the following requirements for JNDI names:

- Do not assign duplicate JNDI names across different resource types (such as mail sessions versus URL configurations).
- Do not assign duplicate JNDI names for multiple resources of the same type in the same scope.

Description
Specifies the description of the resource.

Category
Specifies the category string, which you can use to classify or group the resource.

Specification
Specifies the string from which to form a URL.

Administering resource environment entries

Configuring new resource environment entries to map logical environment resource names to physical names

This topic provides instructions on configuring *new* resource environment entries, which define environment resources that are the binding targets for resource-environment-references in an application's deployment descriptor.

Procedure

1. Configure a resource environment provider, which is a library that provides the implementation for an environment resource factory. In the administration console, begin by clicking **Resources > Resource Environment > Resource Environment Providers > New**. See the topic New Resource Environment Provider for more information.
2. After saving your resource environment provider, go to the Additional Properties heading and click **Resource environment entries**. Click **New** to define a new resource environment entry. Refer to the topic Resource environment entry settings for descriptions of the required fields.
3. You also might need to create a referenceable, which specifies the factory class name that converts information in the name space into a class instance for your resource. To view the appropriate administrative console page for referenceables, click **Resources > Resource Environment > Resource Environment Providers >** *your_resource_environment_provider* > **Referenceables**. Click **New** to begin the configuration process. See the topic Referenceable settings for descriptions of the required fields.

Resource environment providers and resource environment entries

A resource environment reference maps a logical name used by the client application to the physical name of an object.

Not all objects bound into the server JNDI namespace are intended for use by an application client. For example, the WebSphere Application Server client run time does not support the use of Java 2 Connector (J2C) objects on the client. The object needs to be remotable, and the client-side implementations must be made available on the application client run-time classpath.

Resource environment references are different than resource references. Resource environment references allow your application client to use a logical name to look up a resource bound into the server JNDI namespace. A resource reference allows your application to use a logical name to look up a local J2EE resource. The J2EE specification does not specify a particular implementation of a resource.

Resource environment provider collection

Use this page to view resource environment providers, which encapsulate the referenceables that convert resource environment entry data into resource objects.

To view this administrative console page, click **Resources > Resource Environment > Resource Environment Providers**.

Name:

Specifies a text identifier for the resource environment provider.

Data type String

Scope:

Specifies the scope of this resource environment provider, which automatically becomes the scope of the resource environment entries that you define with this provider.

Description:

Specifies a text string describing the resource environment provider.

Data type String

Resource environment provider settings:

Use this page to create settings for a resource environment provider.

To view this administrative console page, click **Resources** > **Resource environment** > **Resource environment providers** > *resource environment provider*.

Scope:

Specifies the scope of this resource environment provider, which automatically becomes the scope of the resource environment entries that you define with this provider.

Name:

Specifies the name of the resource provider.

Data type String

Description:

Specifies a text description for the resource provider.

Data type String

New Resource environment provider:

Use this page to define the configuration for a library that provides the implementation for a environment resource factory.

To view this administrative console page, click **Resources > Resource Environment > Resource Environment Providers > New**.

Scope:

Specifies the scope of this resource environment provider, which automatically becomes the scope of the resource environment entries that you define with this provider.

Name:

Specifies a text identifier for the resource environment provider.

Data type String

Description:

Specifies a text string describing the resource environment provider.

Data type String

Resource environment entries collection

Use this page to view configured resource environment entries. Within an application server name space, the data contained in a resource environment entry is converted into an object that represents a physical resource. This resource is frequently called an *environment resource*.

An environment resource can be of any arbitrary type. See the latest EJB specification for more information about resource environment references and environment resources.

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Environment** > **Resource environment entries**
- **Resources** >**Resource Environment** > **Resource Environment Providers** > *resource_environment_provider* > **Resource Environment Entries**

Name:

Specifies a text identifier that helps distinguish this resource environment entry from others.

For example, you can use *My Resource* for the name.

Data type String

JNDI Name:

Specifies the string to be used when looking up this environment resource using JNDI.

This is the string to which you bind resource environment reference deployment descriptors.

Data type String

Scope:

Specifies the resource environment entry scope, which is inherited from the resource environment provider.

Provider:

Specifies the resource environment provider for this entry. The provider encapsulates the classes that, when implemented, convert resource environment entry data into resource objects.

Description:

Specifies text for information to help further identify and distinguish this resource

Data type String

Category:

Specifies a category you can use to group environment resources according to some common feature.

It is strictly an organizational property and has no effect on the function of the environment resource.

Data type String

Resource environment entry settings:

Use this page to configure resource environment entries. Within an application server name space, the data contained in a resource environment entry is converted into an object that represents a physical resource. Rather than represent a connection factory, which provides connections to a resource, this object *directly* represents a resource. This design can make the resource available to application modules that do not run entirely on the application server. Examples include some application clients and web modules.

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Environment** > **Resource environment entries** > *resource_environment_entry*
- **Resources** >**Resource Environment** > **Resource Environment Providers** > *resource_environment_provider* > **Resource Environment Entries** > *resource_environment_entry*

Scope:

Specifies the scope of the resource environment provider, which is a library that supplies the implementation class for a resource environment factory. Within a JNDI name space, WebSphere Application Server uses the factory to transform your resource environment entry into an object that directly represents a physical resource.

Provider:

Specifies the resource environment provider.

Provider shows all of the existing resource environment providers that are defined at the relevant scope. Select one from the list if you want to use an existing resource environment provider as Provider.

Name:

Specifies a display name for the resource.

Data type String

JNDI name:

Specifies the JNDI name for the resource, including any naming subcontexts.

This name is used as the linkage between the platform's binding information for resources defined by a module's deployment descriptor and actual resources bound into JNDI by the platform.

Data type String

Important: Adhere to the following requirements for JNDI names:

- Do not assign duplicate JNDI names across different resource types (such as resource environment entries versus J2C connection factories).
- Do not assign duplicate JNDI names for multiple resources of the same type in the same scope.

Description:

Specifies a text description for the resource.

Data type String

Category:

Specifies a category string that you can use to classify or group the resource.

Data type String

Referenceables:

Specifies the referenceable, which encapsulates the class name of the factory that converts resource environment entry data into a class instance for a physical resource.

Data type Drop-down menu

Referenceables collection

Use this page to view configured referenceables, which encapsulate the class name of the factory that converts information in the name space into a class instance for a physical resource.

To view this administrative console page, click **Resources** > **Resource environment** > **Resource Environment Providers** > *resource_environment_provider* > **Referenceables**.

Factory Class name:

Specifies a javax.naming.spi.ObjectFactory implementation name

Data type String

Class name:

Specifies the package name of the referenceable, for example: `javax.naming.Referenceable`

Data type String

Referenceables settings:

Use this page to set the class name of the factory that converts information in the name space into a class instance of a physical resource.

To view this administrative console page, click **Resources > Resource Environment > Resource Environment Providers >** *resource_environment_provider* **> Referenceables >** *referenceable*.

Factory class name:

Specifies a javax.naming.ObjectFactory implementation class name

Data type String

Class name:

Specifies the Java type to which a Referenceable provides access, for binding validation and to create the reference.

Data type String

Resource environment references

Use this page to designate how the resource environment references of application modules map to remote resources, which are represented in the product as resource environment entries.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Resource environment references**.

Each row of the table depicts a resource environment reference within a specific module of your application. If you bound any references to resource environment entries during application assembly, you see the JNDI names of those resource environment entries in the applicable rows.

To set the mapping relationships between your resource environment references and resource environment entries:

1. Select a row. Be aware that if you check multiple rows on this page, the resource mapping target that you select in step 2 applies to all of those references.
2. Click **Browse** to select a resource environment entry from the new page that is displayed, the Available Resources page. The Available Resources page shows all resource environment entries that are available mapping targets for your application references.
3. Click **Apply**. The console displays the Resource environment references page again. In the rows that you previously selected, you now see the JNDI name of the new resource mapping target.
4. Repeat the previous steps as necessary.
5. Click **OK**. You now return to the general configuration page for your enterprise application.

Table column heading descriptions:

Select:

Select the check boxes of the rows that you want to edit.

Module:

The name of a module in the application.

EJB:

The name of an enterprise bean that is accessed by the module.

URI:

Specifies location of the module relative to the root of the application EAR file.

Reference binding:

The name of a resource environment reference that is declared in the deployment descriptor of the application module. The reference corresponds to a resource that is bound as a resource environment entry into the JNDI name space of the application server.

JNDI name:

The Java Naming and Directory Interface (JNDI) name of the resource environment entry that is the mapping target of the resource environment reference.

Data type String

Chapter 12. Administering Messaging resources

This page provides a starting point for finding information about the use of asynchronous messaging resources for enterprise applications with WebSphere Application Server.

WebSphere Application Server supports asynchronous messaging based on the Java Message Service (JMS) and the Java EE Connector Architecture (JCA) specifications, which provide a common way for Java programs (clients and Java EE applications) to create, send, receive, and read asynchronous requests, as messages.

JMS support enables applications to exchange messages asynchronously with other JMS clients by using JMS destinations (queues or topics). Some messaging providers also allow WebSphere Application Server applications to use JMS support to exchange messages asynchronously with non-JMS applications; for example, WebSphere Application Server applications often need to exchange messages with WebSphere MQ applications. Applications can explicitly poll for messages from JMS destinations, or they can use message-driven beans to automatically retrieve messages from JMS destinations without explicitly polling for messages.

WebSphere Application Server supports the following messaging providers:

- The WebSphere Application Server default messaging provider (which uses service integration as the provider).
- The WebSphere MQ messaging provider (which uses your WebSphere MQ system as the provider).
- Third-party messaging providers that implement either a JCA Version 1.5 resource adapter or the ASF component of the JMS Version 1.0.2 specification.

Managing messaging with the default messaging provider

The default messaging provider is installed and runs as part of WebSphere Application Server, and is based on service integration technologies. For messaging between application servers, perhaps with some interaction with a WebSphere MQ system, you can configure your messaging applications to use messaging resources provided by the default messaging provider.

Before you begin

For messaging between application servers, perhaps with some interaction with a WebSphere MQ system, you can use the default messaging provider as described in this topic. To integrate WebSphere Application Server messaging into a predominantly WebSphere MQ network, you can use the WebSphere MQ messaging provider. You can also use a third-party messaging provider. To choose the provider that is best suited to your needs, see Choosing a messaging provider.

About this task

The default messaging provider supports JMS 1.1 domain-independent interfaces (sometimes referred to as “unified” or “common” interfaces). This enables applications to use common interfaces for both point-to-point and publish/subscribe messaging. This also enables both point-to-point and publish/subscribe messaging within the same transaction. With JMS 1.1, this approach is recommended for new applications. The domain-specific interfaces are supported for backwards compatibility for applications developed to use domain-specific queue interfaces, as described in section 1.5 of the JMS 1.1 specification.

You can use the WebSphere Application Server administrative console to configure JMS resources for applications, and can manage messages and subscriptions associated with JMS destinations.

WebSphere Application Server Version 5.1 Java EE applications can use messaging resources of the default messaging provider in later versions. This JMS interoperation from WebSphere Application Server Version 5.1 to later versions is enabled and managed by a WebSphere MQ client link created on the node in the later version. This JMS interoperation is only intended as an aid to the migration from the embedded messaging in WebSphere Application Server Version 5.1 to the default messaging provider in the later version.

Note: Java EE applications running under the later version can use messaging resources of Version 5 embedded messaging without any need for a WebSphere MQ client link.

For more information about using the default messaging provider of WebSphere Application Server, see the following topics:

Procedure

- Default messaging
- "Configuring resources for the default messaging provider"
- "Interoperating with a WebSphere MQ network" on page 545
- Migrating from WebSphere Application Server Version 5 embedded messaging
- "Enabling WebSphere Application Server Version 5.1 JMS usage of messaging resources in later versions of the product" on page 583
- "Configuring the messaging engine selection process for JMS applications" on page 593
- "Managing messages and subscriptions for default messaging JMS destinations" on page 595
- "Using JMS from stand-alone clients to interoperate with service integration resources" on page 596
- "Using JMS from a third party application server to interoperate with service integration resources" on page 605

Configuring resources for the default messaging provider

Use the following tasks to configure JMS connection factories, activation specifications, and destinations for the default messaging provider.

About this task

Use these tasks to configure administrative JMS resources provided by the default messaging provider.

These administrative JMS resources are in addition to any temporary JMS destinations created by applications.

- List JMS resources.
- Configure a unified connection factory.
- Configure a queue connection factory.
- Configure a topic connection factory.
- Configure a queue.
- Configure a topic.
- Configure an activation specification.

Listing JMS resources for the default messaging provider

Use the WebSphere Application Server administrative console to list JMS resources for the default messaging provider, for administrative purposes.

About this task

You use the WebSphere Application Server administrative console to list JMS resources, if you want to view, modify or delete any of the following resources:

- Activation specifications

- Unified connection factories
- Queue connection factories
- Topic connection factories
- Queues
- Topics

When you use the administrative console to locate these resources, two different navigation pathways are available:

- **Provider-centric navigation** lets you view all providers, or just those for a specified scope, then navigate to a specific resource for a specific provider. This is the traditional way of navigating to a resource when you know which provider supports it. Any navigation that starts with **Resources -> JMS -> JMS providers** is provider-centric.
- **Resource-centric navigation** lets you view all resources of a specified type, then navigate to a resource. This is useful if you want to find a resource, but you do not know which provider supports it (you can list all resources of a given type across all scopes, for all providers, in a single panel). Any navigation that follows the pattern **Resources -> JMS -> *resource_type*** is resource-centric, where *resource_type* is one of the resource types previously listed.

You can use either of these navigation pathways to locate JMS resources of any type.

Procedure

- Use provider-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console.
  2. In the navigation pane, click **Resources -> JMS -> JMS providers**.

     The JMS providers collection panel is displayed. This lists all currently configured messaging providers across all scopes (you can modify the scope if required).
  3. Select the required JMS provider.

     The settings panel for this provider is displayed. The configuration tab contains a set of links to all the JMS resources owned by this provider.
  4. Click the link for a JMS resource type. For example, click **Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories for this provider.
  5. Select the required queue connection factory.
- Use resource-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console.
  2. In the navigation pane, click **Resources -> JMS -> Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories across all messaging providers.
  3. Select the required queue connection factory.

Results

You can now view and work with the resource properties.

Configuring JMS resources for point-to-point messaging

Configure connection factories, queues and service integration bus destinations for point-to-point messaging.

About this task

For an application to use point-to-point messaging with JMS queues, you configure the following JMS resources. These JMS resources depend on the corresponding configuration of service integration

resources, including a service integration bus and a queue. For more information about defining these resources, see the related tasks.

Procedure

- Configure a connection factory.

  Use the connection factory type that matches the JMS level and domain pattern in which an application is developed. For example, use a domain-independent JMS connection factory for a JMS application developed to use JMS 1.1 domain-independent interfaces, and use a JMS queue connection factory for a JMS application developed to use domain-specific queue interfaces.
- Configure a queue.

  A *JMS queue* is an administrative object that encapsulates the name of a queue destination on a service integration bus. Applications find the JMS queue by looking up its name in the JNDI namespace.
- Configure a queue destination.

  For each JMS queue, define a bus destination of type queue on the appropriate service integration bus.

Configuring JMS resources for publish/subscribe messaging

Use this task to configure a JMS resources for publish/subscribe messaging.

About this task

For an application to use publish/subscribe messaging with JMS topics, you configure the following JMS resources. These JMS resources depend on the corresponding configuration of service integration resources, including a service integration bus and topicspace. For more information about defining these resources, see the related tasks.

Procedure

- A JMS connection factory

  You should use the connection factory type that matches the JMS level and domain pattern in which an application is developed. For example, use a domain-independent JMS connection factory for a JMS application developed to use JMS 1.1 domain-independent interfaces, and use a JMS topic connection factory for a JMS application developed to use domain-specific topic interfaces.

  If durable subscriptions are to be used by the application, set the durable subscription properties on the connection factory.
- A JMS topic

  A *JMS topic* is an administrative object that encapsulates the name of a topic and a topic space on a service integration bus. Applications can obtain the JMS topic by looking its name up in the JNDI namespace.
- Service integration resources For each JMS topic, define a bus destination as a topic space on a service integration bus, and assign the JMS topic to a topic name within that topic space.

Configuring a unified connection factory for the default messaging provider

Use this task to configure a unified JMS connection factory for applications that use the JMS 1.1 domain-independent (unified) interfaces.

About this task

The term “unified” refers to the support of both queues and topics by the same connection factory. This is similar to the JMS 1.1 domain-independent interfaces (referred to as the “common interfaces” in the JMS specification). A unified JMS connection factory can be used for both point-to-point and publish/subscribe JMS messaging. With JMS 1.1, this approach is preferred to the domain-specific queue connection factory and topic connection factory.

This task contains an optional step for you to create a new connection factory if you have not already created the connection factory that you want to configure.

Procedure

1. Display the default messaging provider. In the navigation pane, click **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to configure a unified connection factory.
3. Optional: Change the **Scope** check box to set the level at which the connection factory is to be visible.
4. In the content pane, under the Additional properties heading, click **Connection factories**. This displays any existing connection factories in the content pane.
5. If the connection factory is for use by client applications, display the properties of the JMS connection factory. If you want to display an existing JMS connection factory, click one of the names listed.

   Alternatively, if you want to create a new JMS connection factory, click **New**, then specify the following required properties:

   Name Type the name by which the connection factory is known for administrative purposes.

   JNDI name
   Type the JNDI name that is used to bind the connection factory into the namespace.

   Bus name
   Select the name of the service integration bus to which the connection factory is to create connections. This service integration bus hosts the destinations that the JMS queues and topics represent.
6. Review the other properties for the JMS connection factory, to check that the defaults are suitable.

   If the connection factory is for use by client applications running outside of an application server, specify suitable provider endpoints. For more information about configuring provider endpoints, see “Configuring a connection to a non-default bootstrap server” on page 531.

   By default, connections created by using the connection factory in the server containers (for example, from an enterprise bean) are pooled by using Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection pooling. You can modify the connection pool settings for the connection factory by selecting the Connection pool properties link in the Additional Properties section of the panel. For more information about changing the connection pool properties, see Changing connection pool settings with the wsadmin tool.
7. Click **OK**.
8. Save your changes to the master configuration.

Administrative properties for JMS connections to a bus:

You can configure properties to enable workload management of connections to a service integration bus for JMS applications. The same properties can also be used to control the client connection topology. For example, connection options can be specified such that client applications only connect to a set of client serving messaging engines and never to the set of destination serving messaging engines in a bus.

The properties for connecting JMS applications to a bus are used by the administrator. The JMS applications do not specify how to connect to the bus, beyond using a JMS connection factory or JMS activation specification (for message-driven beans).

The general aim of connecting to a bus is to connect to a suitable messaging engine that provides the message point for a JMS destination that the application is to use. Applications running inside an application server can locate a suitable messaging engine and connect directly to the selected messaging engine. Client applications running outside of an application server cannot locate a suitable messaging engine themselves, these clients must use a bootstrap server to locate a suitable messaging engine on behalf of the client application.

When an application connects to the bus, the bus chooses a suitable messaging engine based on administrative properties of the JMS connection factory or activation specification that the application uses. For maximum connection flexibility, you can leave most properties to default, the only required connection property is the name of the bus that the application is to connect to.

The bus uses the following general process to choose a suitable messaging engine, based on the value you select for the Connection proximity property. If you understand this process, you can better configure the properties that control how the bus chooses messaging engines.

- If a **Target group** is specified then the process checks the nearest messaging engine that supports the required **Remote transport chain** and is a member of the target group in the bus. If the messaging engine is within the specified **Connection proximity** it is chosen as a suitable messaging engine for the application to connect to.
- If a **Target group** is not specified then the process checks the nearest messaging engine that supports the required **Remote transport chain** in the bus. A messaging engine in the same server is nearer than a messaging engine in the same host, which is nearer than a messaging engine in another host. If the messaging engine is within the specified **Connection proximity** it is chosen as a suitable messaging engine for the application to connect to.
- If the selected messaging engine is not within the specified **Connection proximity**, then the **Target significance** is used. If the **Target significance** is set to Required, then no connection is possible and the connection request is rejected with no suitable messaging engine being available. If the **Target significance** is set to Preferred then the target group is ignored and the nearest messaging engine that supports the required **Remote transport chain** is used. If no messaging engine is found then the connection request is rejected with no suitable messaging engine being available.

The following rules are used to test the connection proximity for a selected messaging engine:

- If the Connection proximity value is Bus, then the selected messaging engine is used.
- If the Connection proximity value is Host, and the selected messaging engine is in the same host as the application (or bootstrap server), then the selected messaging engine is used. Otherwise, one of the following options is chosen.
  - If the selected messaging engine is not in the same host as the application (or the bootstrap server), and the **Target significance** is set to Required, then no connection is possible and the connection request is rejected with no suitable messaging engine being available.
  - If the **Target significance** is set to Preferred then the nearest messaging engine - in the same host - that supports the required **Remote transport chain** is used
  - If no suitable messaging engine is found, then the connection request is rejected.
- If the Connection proximity value is Server, and the selected messaging engine is in the same server as the application (or bootstrap server), then the selected messaging engine is used. Otherwise, one of the following options is chosen.
  - If the selected messaging engine is not in the same server as the application (or is in the bootstrap server), and the **Target significance** is set to Required, then no connection is possible and the connection request is rejected with no suitable messaging engine being available.
  - If the **Target significance** is set to Preferred then the nearest messaging engine - in the same server - that supports the required **Remote transport chain** is used
  - If no suitable messaging engine is found, then the connection request is rejected.

When a connection is made to a messaging engine in the same server as the application, the connection is made directly through memory, so the **Remote transport chain** is ignored.

Configuring a queue connection factory for the default messaging provider

Use this task to configure a JMS queue connection factory for point-to-point messaging with the default messaging provider. This is intended more for backwards compatibility, as described in section 1.5 of the JMS 1.1 specification.

About this task

To configure a JMS queue connection factory for the default messaging provider, use the administrative console to complete the following steps. This task contains an optional step for you to create a new queue connection factory.

Procedure

1. Display the default messaging provider. In the navigation pane, click **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to configure a queue connection factory.
3. Optional: Change the **Scope** check box to set the level at which the connection factory is to be visible, according to your needs.
4. In the content pane, under the Additional properties heading, click **Queue connection factories**. This displays any existing JMS queue connection factories for the default messaging provider in the content pane.
5. Display the properties of the JMS connection factory. If you want to display an existing JMS connection factory, click one of the names listed.

   Alternatively, if you want to create a new JMS connection factory, click **New**, then specify the following required properties:

   Name Type the name by which the connection factory is known for administrative purposes.

   JNDI name
   Type the JNDI name that is used to bind the connection factory into the namespace.

   Bus name
   Type the name of the service integration bus that the connection factory is to create connections to. This service integration bus hosts the destinations that the JMS queues and topics represent.
6. Review the other “Default messaging provider queue connection factory [Settings]” on page 641, to check that the defaults are suitable.

   If the connection factory is for use by client applications running outside of an application server, specify suitable provider endpoints. For more information about configuring provider endpoints, see “Configuring a connection to a non-default bootstrap server” on page 531.

   By default connections created by using the connection factory in the server containers (for example, from an enterprise bean) are pooled by using Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection pooling. You can modify the connection pool settings for the connection factory by selecting the Connection pool properties link in the Additional Properties section of the panel. For more information about changing the connection pool properties, see Changing connection pool settings with the wsadmin tool.
7. Click **OK**.
8. Save your changes to the master configuration.

Configuring a topic connection factory for the default messaging provider

Use this task to configure a JMS topic connection factory for publish/subscribe messaging with the default messaging provider. This is intended more for backwards compatibility, as described in section 1.5 of the JMS 1.1 specification.

About this task

You configure JMS connection factories when you deploy JMS applications that use publish/subscribe messaging.

This task contains an optional step for you to create a new topic connection factory.

Procedure

1. Display the default messaging provider. In the navigation pane, click **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to configure a topic connection factory.
3. Optional: Change the **Scope** check box to set the level at which the connection factory is to be visible, according to your needs.
4. In the content pane, under the Additional properties heading, click **Topic connection factories**. This displays any existing JMS topic connection factories for the default messaging provider in the content pane.
5. Optional: Display the properties of the JMS connection factory. If you want to display an existing JMS connection factory, click one of the names listed.

   Alternatively, if you want to create a new JMS connection factory, click **New**, then specify the following required properties:

   Name Type the name by which the connection factory is known for administrative purposes.

   JNDI name
   Type the JNDI name that is used to bind the connection factory into the namespace.

   Bus name
   Type the name of the service integration bus that the connection factory is to create connections to. This service integration bus hosts the destinations that the JMS queues and topics represent.
6. Review the other properties for the connection factory, to check that the defaults are suitable.

   If the connection factory is for use by client applications running outside of an application server, specify suitable provider endpoints. For more information about configuring provider endpoints, see "Configuring a connection to a non-default bootstrap server" on page 531.

   By default, connections created by using the connection factory in the server containers (for example, from an enterprise bean) are pooled by using Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection pooling. You can modify the connection pool settings for the connection factory by selecting the Connection pool properties link in the Additional Properties section of the panel. For more information about changing the connection pool properties, see Changing connection pool settings with the wsadmin tool.
7. Click **OK**.
8. Save your changes to the master configuration.

Configuring a queue for the default messaging provider

Use this task to configure a JMS queue for point-to-point messaging with the default messaging provider.

Before you begin

This task configures a JMS queue to use a queue destination on a service integration bus. The queue destination is the virtual location on the bus where messages are stored and processed for the JMS queue. If you have not created the required destination, use the task described in Configuring a destination for JMS queues.

About this task

To configure a JMS queue for the default messaging provider, use the administrative console to complete the following steps.

Procedure

1. Display the default messaging provider. In the navigation pane, click **Resources -> JMS -> JMS providers**.

2. Select the default provider for which you want to configure a queue.
3. Optional: Change the **Scope** check box to set the level at which the JMS queue is to be visible, according to your needs.
4. In the content pane, under the Additional properties heading, click **Queues**. This displays any existing JMS queues for the default messaging provider in the content pane.
5. Display the properties of the JMS queue. If you want to display an existing JMS queue, click one of the names listed.

   Alternatively, if you want to create a new JMS queue, click **New**, then specify the following required properties:

   Name Type the name by which the queue is known for administrative purposes.

   JNDI name
   Type the JNDI name that is used to bind the queue into the namespace.

   Queue name
   Select the name of the queue on a service integration bus that this JMS queue is to use.
6. Specify properties for the JMS queue, according to your needs.
7. Click **OK**.
8. Save your changes to the master configuration.

Configuring a topic for the default messaging provider

Use this task to configure a JMS topic for publish/subscribe messaging with the default messaging provider.

Before you begin

This task configures a JMS topic to use a topic space destination on a service integration bus. The topic space (a hierarchical collection of topics) is the virtual location on the bus where messages are stored and processed for the JMS topic. If you have not created the required destination, use the task described in Configuring a destination for publish/subscribe messaging.

About this task

To configure a JMS topic for the default messaging provider, use the administrative console to complete the following steps. This task contains an optional step for you to create a new JMS topic.

Procedure

1. Display the default messaging provider. In the navigation pane, expand **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to configure a topic.
3. Optional: Change the **Scope** check box to set the level at which the topic is to be visible, according to your needs.
4. In the content pane, under the Additional properties heading, click **Topics**. This displays any existing JMS topics for the default messaging provider in the content pane.
5. Display the properties of the JMS topic. If you want to display an existing JMS topic, click one of the names listed.

   Alternatively, if you want to create a new JMS topic, click **New**, then specify the following required properties:

   Name Type the name by which the topic is known for administrative purposes.

   JNDI name
   Type the JNDI name that is used to bind the topic into the namespace.

Topic name
Type the name of the topic (as a qualifier in the topic space) that this JMS topic is to use.

Topic space
Type the name of the topic space that this JMS topic is to use.

6. Specify properties for the JMS topic, according to your needs.
7. Click **OK**.
8. Save your changes to the master configuration.

Configuring an activation specification for the default messaging provider

Configure a JMS activation specification to enable a message-driven bean to communicate with the default messaging provider.

About this task

You create a JMS activation specification if you want to use a message-driven bean to communicate with the default messaging provider through Java EE Connector Architecture (JCA) 1.5. JCA provides Java connectivity between application servers such as WebSphere Application Server, and enterprise information systems. It provides a standardized way of integrating JMS providers with Java EE application servers, and provides a framework for exchanging data with enterprise systems, where data is transferred in the form of messages.

One or more message-driven beans can share a single JMS activation specification.

Because a JMS activation specification is a group of messaging configuration properties not a component, it cannot be manually started and stopped. For this reason, to prevent a message-driven bean from processing messages you must complete the following tasks:

- Stop the application that contains the message-driven bean.
- Stop the messaging engine.

All the activation specification configuration properties apart from **Name**, **JNDI name**, **Destination JNDI name**, and **Authentication alias** are overridden by appropriately named activation-configuration properties in the deployment descriptor of an associated EJB 2.1 or later message-driven bean. For an EJB 2.0 message-driven bean, the **Destination type**, **Subscription durability**, **Acknowledge mode** and **Message selector** properties are overridden by the corresponding elements in the deployment descriptor. For either type of bean the **Destination JNDI name** property can be overridden by a value specified in the message-driven bean bindings.

Procedure

1. Start the administrative console.
2. Display the default messaging provider. In the navigation pane, expand **Resources -> JMS -> JMS providers**.
3. Select the default provider for which you want to configure an activation specification.
4. Optional: Change the **Scope** check box to the scope level at which the activation specification is to be visible to applications, according to your needs.
5. In the content pane, under the Additional properties heading, click **Activation specifications**. This lists any existing JMS activation specifications for the default messaging provider in the content pane.
6. Display the properties of the JMS activation specification. If you want to display an existing activation specification, click one of the names listed.

   Alternatively, if you want to create a new activation specification, click **New**, then specify the following required properties:

   "Name" on page 618
   Type the name by which the activation specification is known for administrative purposes.

"JNDI name" on page 618
Type the JNDI name that is used to bind the activation specification into the JNDI namespace.

"Destination type" on page 618
Whether the message-driven bean uses a queue or topic destination.

"Destination JNDI name" on page 618
Type the JNDI name that the message-driven bean uses to look up the JMS destination in the JNDI namespace.

Select the type of destination on the Destination type property.

"Bus name" on page 619
The name of the bus to connect to.

Specify the name of the service integration bus to which connections are made. This must be the name of the bus on which the bus destination identified by the "Destination JNDI name" on page 618 property is defined.

You can either select an existing bus or type the name of another bus. If you type the name of a bus that does not exist, you must create and configure that bus before the activation specification can be used.

7. Specify properties for the JMS activation specification, according to your needs.
8. Optional: Specify the JMS activation specification connection properties that influence how the default messaging provider chooses the messaging engine to which your message-driven bean application connects. By default, the environment automatically connects applications to an available messaging engine on the bus. However you can specify extra configuration details to influence the connection process; for example to identify special bootstrap servers, or to limit connection to a subgroup of available messaging engines, or to improve availability or performance, or to ensure sequential processing of messages received. For information about why and how to do this, see How JMS applications connect to a messaging engine on a bus.
9. Click **OK**.
10. Save your changes to the master configuration.

Deleting JMS resources for the default messaging provider

Use this task with the WebSphere Application Server administrative console to delete JMS resources.

About this task

To delete JMS resources, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to delete resources.
3. In the content pane, under the Additional properties heading, select the link for the type of JMS resource:
   - Connection factory
   - Queue connection factory
   - Topic connection factory
   - Queue
   - Topic
   - Activation specification

   This displays a list of the selected JMS resource type in the content pane.
4. Select the check box next to the JMS resource that you want to delete.

5. Click **Delete**
6. Save your changes to the master configuration.

Configuring JMS connection factory properties for durable subscriptions

Use this task to configure durable subscription properties of JMS connection factories for use by enterprise beans with the default messaging provider.

About this task

To enable applications to create durable subscriptions to JMS topics with the default messaging provider, you can set a number of properties on JMS connection factories.

If applications use message-driven beans to create durable subscriptions, you should set the properties on the JMS activation specification used by the message-driven beans, as described in Configuring JMS activation specifications for durable subscriptions, instead of as described in this topic.

This topic describes the setting of properties on a unified JMS connection factory. You can also set the same properties on a JMS topic connection factory instead.

To configure the durable subscription properties to a topic for use by enterprise beans with the default messaging provider, use the administrative console to complete the following steps:

Procedure

1. Display the default messaging provider. In the navigation pane, click **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to configure connection factory properties.
3. Optional: Change the **Scope** check box to set the level at which the connection factory is to be visible, according to your needs.
4. In the content pane, under the Additional properties heading, click **Connection factories**. This displays any existing JMS connection factories for the default messaging provider in the content pane.
5. Click the name of the connection factory you want to configure. This displays the properties for the connection factory in the content pane.
6. Specify the following properties for the connection factory:

   Client identifier
   This is the JMS client identifier that applications use to identify durable topic subscriptions created on all connections that use this connection factory. For more information about client identifiers, see section 4.3.2 of the JMS 1.1 specification.

   Durable subscription home
   The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS connection factory.

   This identifies the messaging engine where durable subscriptions are localized on the service integration bus. Administrators can manage the runtime state of durable subscriptions through publication points for that messaging engine.
7. Click **OK**.
8. Save your changes to the master configuration.

What to do next

When applications have created durable subscriptions, you can use the administrative console to manage the runtime state of those subscriptions, as described in “Administering durable subscriptions” on page 2002.

The JMS connection factory has some other advanced properties that you can configure to change the behavior for durable subscriptions. You should not usually need to change these properties from their default values.

Read ahead
This controls read ahead optimization during message delivery. This defines whether the provider can stream messages to durable subscribers ahead of their requests (to provide a performance enhancement).

Share durable subscriptions
This controls whether durable subscriptions can be accessed simultaneously by several subscribers.

If you want to control read ahead optimization during message delivery for individual topics, you can set the Read ahead property on the topics.

Configuring JMS activation specification properties for durable subscriptions

Use this task to configure durable subscription properties of JMS activation specifications for use by message-driven beans with the default messaging provider.

About this task

To enable a message-driven bean (MDB) application to create durable subscriptions on JMS topics with the default messaging provider, you set a number of properties on the JMS activation specification used by the application.

If applications use message-driven beans to create durable subscriptions, you should set the properties on the JMS activation specification used by the message-driven beans, as described in this topic. Otherwise, for enterprise beans to create durable subscriptions, you should set the properties on the JMS connection factory as described in “Configuring JMS connection factory properties for durable subscriptions” on page 526.

Note: The *server_name*-durableSubscriptions.ser file in the *WAS_HOME*/temp directory is used by the messaging service to keep track of durable subscriptions for message-driven beans. If you uninstall an application that contains a message-driven bean, this file is used to unsubscribe the durable subscription. If you have to delete the *WAS_HOME*/temp directory or other files in it, ensure that you preserve this file.

To configure the durable subscription properties to a topic for use by message-driven beans with the default messaging provider, use the administrative console to complete the following steps

Procedure

1. Display the default messaging provider. In the navigation pane, click **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to configure activation specification properties.
3. Optional: Change the **Scope** check box to set the level at which the connection factory is to be visible, according to your needs.
4. In the content pane, under the Additional properties heading, click **Activation specifications**. This displays any existing JMS activation specifications for the default messaging provider in the content pane.
5. Click the name of the activation specification you want to configure. This displays the properties for the activation specification in the content pane.
6. Specify the following properties for the activation specification:

   Client identifier
   This is the JMS client identifier that applications use to identify durable topic subscriptions

created on all connections tht use this activation specification. For more information about client identifiers, see section 4.3.2 of the JMS 1.1 specification.

Durable subscription home
The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS activation specification. This is a required field when using a durable topic subscription.

This identifies the messaging engine where durable subscriptions are localized on the service integration bus. Administrators can manage the runtime state of durable subscriptions through publication points for that messaging engine.

Subscription durability
To be able to create durable subscriptions, set this property to Durable.

Subscription name
The subscription name needed for durable topic subscriptions. Required field when using a durable topic subscription.

Each JMS durable subscription is identified by a subscription name (specified on this property). A JMS connection also has an associated client identifier (specified on the Client identifier property), which is used to associate a connection and its objects with the list of messages (on the durable subscription) that is maintained by the JMS provider for the client.

This subscription name must be unique within a given client identifier.

7. Specify the properties for the activation specification, according to your needs.
8. Click **OK**.
9. Save your changes to the master configuration.

What to do next

When applications have created durable subscriptions, you can use the administrative console to manage the runtime state of those subscriptions, as described in “Administering durable subscriptions” on page 2002.

Configuring shared durable subscriptions for an activation specification:

Use this task to configure the **Share durable subscriptions** option; an attribute of the JMS activation specifications that message-driven beans use with the default messaging provider.

About this task

The **Share durable subscriptions** option controls whether durable subscriptions are shared between subscribers in a cluster. To set this option, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Resources -> JMS -> JMS providers**.
2. Select the default provider for which you want to configure activation specification properties.
3. Optional: Change the **Scope** check box to set the level at which the connection factory is visible, according to your needs.
4. In the content pane, under Additional Properties, click **Activation specifications**.
5. Click the name of the activation specification you want to configure. The properties for the activation specification are displayed in the content pane.
6. Under General Properties, in the Advanced section, set the **Share durable subscriptions** property. Select one of the following options from the list:

In cluster
Clients that are connected to the bus in a cluster member can use the same client identifier and durable subscription name, and can retrieve messages from the durable subscription.

Always shared
All clients, regardless of where they are connected to the bus, can use the same client identifier and durable subscription name, and can retrieve messages from the durable subscription.

Never shared
Clients cannot use the same client identifier and durable subscription name as an existing session.

See the administrative console help for information about the other fields on this page.

7. Click **OK**.
8. Save your changes to the master configuration.

Enabling a provider to stream messages to cloned durable subscriptions

Use this task to enable a provider to stream messages to consumers ahead of their message requests. This is most often used by publish/subscribe consumers to provide a performance enhancement.

About this task

To indicate that a provider must stream messages to consumers, you can set the Read ahead property to Enabled. This property can be set on a connection factory to specify the behavior for all connections created using that connection factory. The property can also be set on JMS topics, to enable different behavior when sending messages to different JMS topics from the same connection.

You are recommended to leave this property set to Default, which enables the messaging provider to decide whether it should stream messages to consumers. The messaging provider makes this decision based on the environment in which the durable subscriber is running. You should only set this property to enable message streaming if you are sure that a durable subscription is used by only one consumer at a time.

- In a non-cloned environment, the default setting enables messaging streaming for durable subscribers.
- For cloned durable subscribers (that is, a durable subscriber that is part of an application cloned in a server cluster), the default setting prevents message streaming and the messages on the subscription are shared among the clones. If you are moving from a non-cloned environment, to cloned durable subscribers, you might see a drop in performance. If you are sure that a durable subscription is still used by only one consumer at a time, you can enable message streaming as described in this topic.

  Server clusters can be used as bus members only in WebSphere Application Server environments that support server clusters.

Messages that are streamed to the consumer but are not consumed before the consumer disconnects are unlocked when the consumer closes. Only then do those messages become available for consumption by other consumers.

To force the messaging provider to stream messages to cloned durable subscriptions, use the administrative console to complete the following steps to change the connection factory:

Procedure

1. Display the JMS connection factory; for example, as described in “Configuring a unified connection factory for the default messaging provider” on page 518. All clones of a durable subscriber use the same JMS connection factory.
2. Set the Read ahead property to Enabled.
3. Click **OK**.
4. Save your changes to the master configuration.

Enabling CMP entity beans and messaging engine data stores to share database connections

Use this task to enable container-managed persistence (CMP) entity beans to share the database connections used by the data store of a messaging engine. Performing this task has been estimated to provide a potential performance improvement of 15% for overall message throughput, but can only be used for entity beans connected to the application server that contains the messaging engine.

About this task

To enable CMP entity beans to share the database connections used by the data store of a messaging engine, complete the following steps.

Procedure

1. Configure the data store to use a data source that is not XA-capable. For more information about configuring a data store, see “Configuring a JDBC data source for a messaging engine” on page 1960.
2. Select the **Share data source with CMP** option. This option is provided on the JMS connection factory or JMS activation specification used to connect to the service integration bus that hosts the bus destination that is used to store and process messages for the CMP bean.

   For example, to select the option on a unified JMS connection factory, complete the following steps:

   a. Display the default messaging provider. In the navigation pane, click **Resources -> JMS -> JMS providers**.
   b. Select the default provider for which you want to configure a unified connection factory.
   c. Optional: Change the **Scope** check box to set the level at which the connection factory is to be visible, according to your needs.
   d. In the content pane, under Additional Properties, click **Connection factories**.
   e. Optional: To create a new unified JMS connection factory, click **New**.

      Specify the following properties for the connection factory:

      Name Type the name by which the connection factory is known for administrative purposes.

      JNDI name
      Type the JNDI name that is used to bind the connection factory into the namespace.

      Bus name
      Type the name of the service integration bus that the connection factory is to create connections to. This service integration bus hosts the destinations that the JMS queues and topics are assigned to.

   f. Optional: To change the properties of an existing connection factory, select its name from one of the connection factories displayed. The properties for the connection factory are displayed in the content pane.
   g. Select the check box for the **Share data source with CMP** field.
   h. Click **OK**.
   i. Save your changes to the master configuration.

   The JMS connection factory can only be used to connect to a “local” messaging engine that is in the application server on which the CMP beans are deployed.
3. Deploy the CMP beans onto the application server that contains the messaging engine, and specify the same data source as that used by the messaging engine. You can use the administrative console to complete the following steps:

   a. Optional: To determine the data source used by the messaging engine, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engines -> *engine_name* -> [Additional Properties] Message store**.

      The **Data source name** field displays the name of the data source, which is by default:

      `jdbc/com.ibm.ws.sib/`*`engine_name`*

b. Click **Applications -> New Application -> New Enterprise Application**.
c. On the first Preparing for the application installation page, specify the full path name of the source application file (.ear file, otherwise known as an EAR file), then click **Next**.
d. On the second Preparing for the application installation page, complete the following steps:
   1) Select the check box for **Generate Default Bindings**. Data source bindings (for EJB 1.1 JAR files) are generated based on the JNDI name, data source, user name, and password options. This results in default data source settings for each EJB JAR file. No bean-level data source bindings are generated.
   2) Under Connection Factory Bindings, select the check box for **Default connection factory bindings:**, then type the JNDI name for the data source and optionally select a **Resource authorization** value.
   3) Click **Next** to display the Install New Application pages. The contents of the application that you are installing determines which pages are available.

4. If your application uses EJB modules that contain CMP beans that are based on the EJB 1.x specification, for **Map default data sources for modules containing 1.x entity beans**, specify a JNDI name for the default data source for the EJB modules. The default data source for the EJB modules is optional if data sources are specified for individual CMP beans.
5. If your application has CMP beans that are based on the EJB 1.x specification, for **Map data sources for all 1.x CMP**, specify a JNDI name for data sources to be used for each of the 1.x CMP beans. The data source attribute is optional for individual CMP beans if a default data source is specified for the EJB module that contains CMP beans.
6. Click **Finish**. If neither a default data source for the EJB module nor a data source for individual CMP beans are specified, a validation error displays and the installation is cancelled.
7. Complete other pages as needed.
8. On the Summary page, verify the cell, node, and server onto which the application modules will install.
   a. Beside **Cell/Node/Server**, click **Click here**.
   b. Verify the settings on the Map modules to servers page that is displayed. Ensure that the application server that is specified contains the messaging engine and its data store.
   c. Specify the web servers as targets that will serve as routers for requests to this application. This information is used to generate the plug-in configuration file (plugin-cfg.xml) for each web server.
   d. Return to the Summary page.
   e. Click **Finish**.

Results

For more information about installing applications, see Installing enterprise application files with the console.

Configuring a connection to a non-default bootstrap server

A bootstrap server is an application server running in the same cell, specifically the same core group, as the service integration bus.

About this task

Connection to a non-default bootstrap server is provided by a JMS connection factory or a JMS activation specification. The connection allows applications to use a bootstrap server with a non-default endpoint address. The provider endpoint syntax example described in this topic is also relevant to bootstrap endpoint configuration in other tasks, for example when configuring a service integration bus link.

To use JMS destinations of the default messaging provider, an application or message-driven bean connects to a messaging engine on the target service integration bus on which the destinations are assigned. For example, a JMS queue is assigned to a queue destination on a service integration bus.

Applications that are running in a server that is part of the same cell as the service integration bus can usually connect to a messaging engine on that bus without requiring provider endpoints to be configured. If the cell has been divided into two core groups, each defined with its own policies, client applications that are running in a client container and client applications that are running outside the WebSphere Application Server environment, cannot automatically locate the required service integration bus so that, unless a core group bridge has been configured between the core groups in the same cell, you must configure one or more provider endpoints. Similarly, unless a core group bridge has been established between the two cells, an application that is running on a server in one cell cannot connect to a bus in another cell without the configuration of provider endpoints.

In the scenarios where provider endpoints are required, the clients or the servers in another bus must complete a bootstrap process through a bootstrap server. The bootstrap server does not have to be a member of the service integration bus, and it does not have to contain any messaging engines. For the application to locate the required bootstrap server, you must configure the provider endpoint property of the JMS connection factory or JMS activation specification used by the client application. When the bootstrap server receives the client request, it selects a messaging engine that matches the criteria specified by the connection factory or activation specification, for example the target transport chain, target group, or connection proximity. It returns the location information for this messaging engine to the client, and the client creates a new connection to the target messaging engine, if necessary.

The following figure shows a client application running outside an application server.

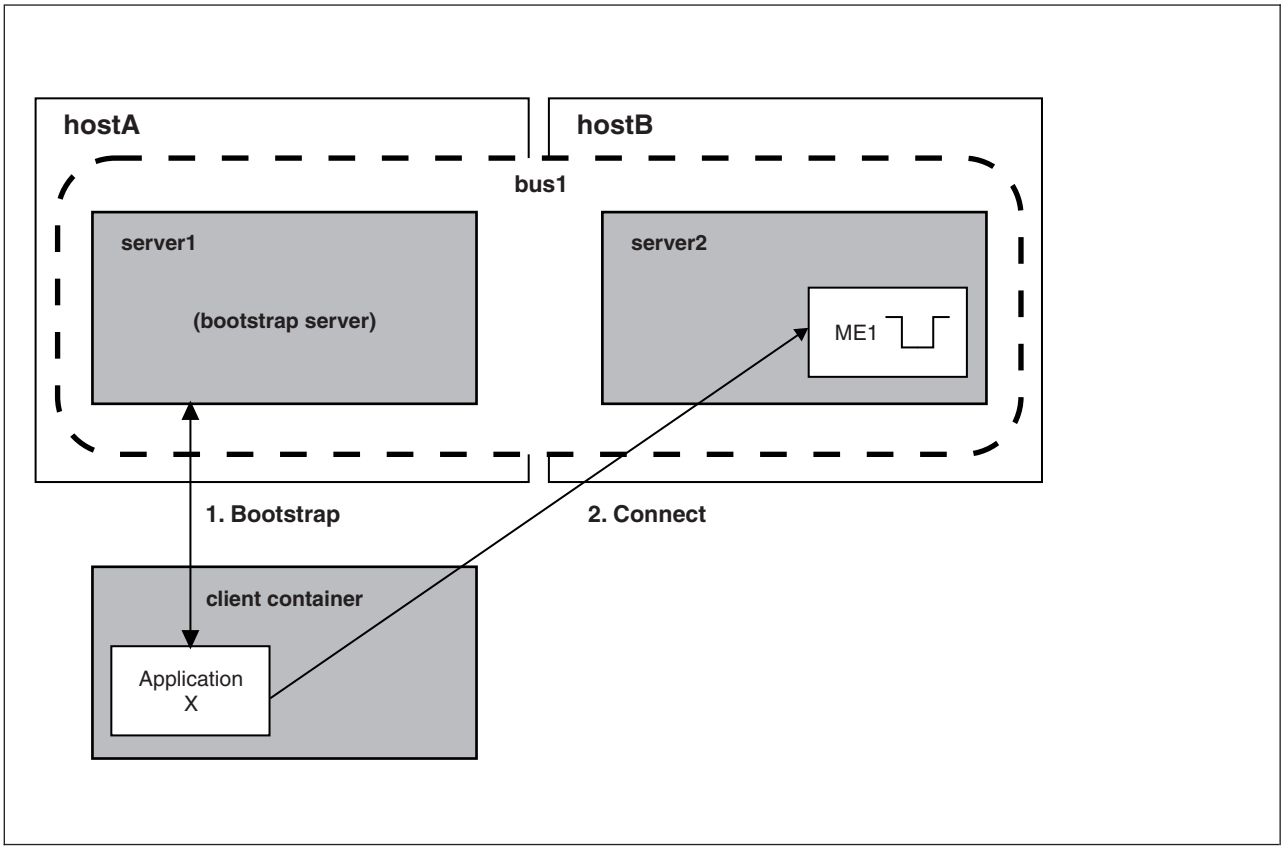


Figure 1. Connection to a messaging engine: Applications running outside an application server

To connect to a messaging engine, the application connects first to a bootstrap server. The bootstrap server selects a messaging engine then tells the client application to connect to that messaging engine.

The following figure shows a message-driven bean running in an application server that is in a different cell to the bus that the message-driven bean needs to be connected to in order to receive messages.

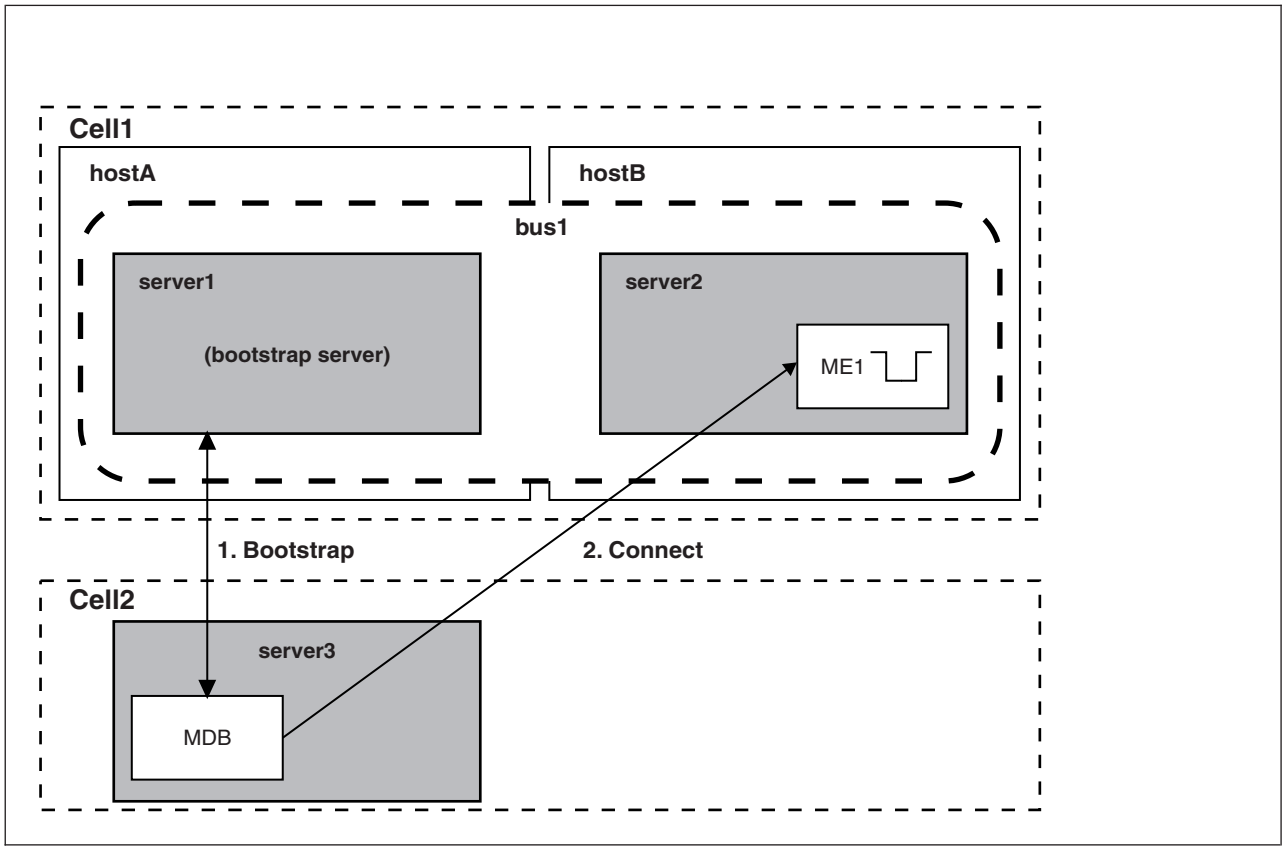


Figure 2. Connection to a messaging engine: message-driven bean application connecting to a destination in a different cell

To connect to a messaging engine, the message-driven bean connects first to a bootstrap server. The bootstrap server selects a messaging engine then tells the message-driven bean to connect to that messaging engine.

A bootstrap server listens on an endpoint that is defined by the combination of:

- The host name of the host on which the bootstrap server is running
- A specific port that is either SIB_END_POINT or, if security is enabled, SIB_ENDPOINT_SECURE_ADDRESS
- A bootstrap transport chain

JMS connection factory properties control how an application connects to a messaging engine, and which messaging engine is selected. If you deploy the application to an application server on which the service integration bus (SIB) service is enabled, the system uses the SIB service to locate a messaging engine that matches the connection factory criteria. The SIB service is aware of all the messaging engines running on servers in the core group of which the application server to which the application is deployed is a member.

If a suitable messaging engine is found, the application is connected to it, and does not use any provider endpoints specified on the connection factory.

Note: This means that you cannot deploy an application to one cell to connect to a bus with the same name in a different cell. Instead the application connects to the bus in the local cell.

The provider endpoints from the connection factory are used to connect to a remote bootstrap server if any of the following conditions are true:

- The application is running as a client application outside of an application server.
- No SIB service is running in the application server to which the application is deployed.
- The SIB service cannot find a suitable messaging engine for the application to connect to.

If you do not specify a value for the provider endpoints in the connection factory, the default value depends on whether the application has supplied a password.

- If the application does not supply a password, a default endpoint address of localhost:7276:BootstrapBasicMessaging is used. That is, by default, applications try to use a bootstrap server on the same host as the client, using port 7276 and the predefined bootstrap transport chain called BootstrapBasicMessaging.
- If the application does supply a password, the default secure port of 7286 and the transport chain BootstrapSecureMessaging is used to prevent the transmission of an unencrypted password to the server.

Note: For the IBM i platform, you must (at least) change the default host name from `localhost` to *your.server.name*.

If you want an application to use a bootstrap server with a different endpoint address, you must specify the required endpoint address on the **Provider endpoints** property of the JMS connection factories or JMS activation specifications that the client application or message-driven bean uses. You can specify one or more endpoint addresses of bootstrap servers by using a comma-separated list.

The endpoint addresses for bootstrap servers must be specified in every JMS connection factory that is used by applications outside of an application server. To avoid having to specify a long list of bootstrap servers, you can provide a few highly-available servers as dedicated bootstrap servers. Then you can specify a short list of bootstrap servers on each connection factory.

This task is based on an application that uses a unified JMS connection factory. You can use the same task to configure a JMS queue connection factory or JMS topic connection factory, but during the task you must select the appropriate type of connection factory instead of a JMS queue connection factory. You can also use this task to configure a JMS activation specification instead of a JMS connection factory.

When you configure a connection to a non-default bootstrap server, specify the required values and use colons as separators. The syntax is as follows:

```
[ [host_name] [ ":" [ port_number] [ ":" chain_name] ] ]
```

Specifying host_name : chain_name instead of host_name : : chain_name (with two colons) is incorrect. The default value applies if you do not specify a value, but you must separate the fields with ":"s.

For an application to use a bootstrap server with a non-default endpoint address, complete the following steps.

Procedure

1. Identify the endpoint address of the application server that you want to use as the bootstrap server. The endpoint address has the form *host_name*:*port_number*:*chain_name*.

   host_name
   The name of the host on which the server runs. It can be an IP address. For an IPv6 address, put square braces ([]) around *host_name*.The default is `localhost`.

   Note: You must (at least) change the default host name from `localhost` to *your.server.name*.

port_number
Where specified, one of the following addresses of the messaging engine hosting the remote end of the link:

- If security is not enabled: SIB_ENDPOINT_ADDRESS
- If security is enabled, for secure connections: SIB_ENDPOINT_SECURE_ADDRESS

This value is mandatory. The default is 7276 if the application has not specified a password, or 7286 if a password has been specified.

To find either of the *port_number* values by using the administrative console, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Communications] Ports**.

chain_name
The name of a predefined bootstrap transport chain used to connect to the bootstrap server. If not specified, the default is BootstrapBasicMessaging if a password has not been provided, or BootstrapSecureMessaging if a password has been provided.

The following predefined bootstrap transport chains are provided:

BootstrapBasicMessaging
The server transport chain InboundBasicMessaging (JFAP-TCP/IP).

BootstrapSecureMessaging
The server transport chain InboundSecureMessaging (JFAP-SSL-TCP/IP).

BootstrapTunneledMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. To do this, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**. This transport chain tunnels JFAP and uses HTTP wrappers.

BootstrapTunneledSecureMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. To do this, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**. This transport chain tunnels JFAP and uses HTTP wrappers.

If you want to provide more than one bootstrap server, identify all the required endpoint addresses. Separate each endpoint address with a comma. You should be able to specify the endpoint address for each bootstrap server; for example, for a server assigned non-secure port 7278, on host boothost1, and using the default transport chain BootstrapBasicMessaging:

```
boothost1:7278:BootstrapBasicMessaging
```

or

```
boothost1:7278
```

or, for a server assigned secure port 7289, on host boothost2, and using the predefined transport chain BootstrapTunneledSecureMessaging:

```
boothost2:7289:BootstrapTunneledSecureMessaging
```

2. Optional: Configure the endpoint address of the bootstrap server on the Provider endpoint property of the connection factory.

   If the client application uses a JMS connection factory in the client container, use the Application Client Resource Configuration tool (ACRCT).

   a. Start the tool and open the EAR file for which you want to configure the JMS connection factory. The EAR file contents are displayed in a tree view.
   b. From the tree, select the JAR file in which you want to configure the JMS connection factory.

c. Expand the JAR file to view its contents.
d. Expand **Messaging Providers > Default Provider > Connection Factories**.
e. Display the general properties of the connection factory.
   - To use an existing JMS connection factory, click the name of the connection factory.
   - To create a new JMS connection factory, click **New**.

   For more information about configuring a JMS connection factory in the JMS provider configuration for your application client, see "Configuring Java messaging client resources" on page 72.
f. On the **General** tab, ensure that the **Provider Endpoints** property includes the provider endpoint address for each bootstrap server. Type the value as a comma-separated list of endpoint addresses, for example:

   ```
   boothost1:7278,boothost2:7289:BootstrapTunneledSecureMessaging
   ```
g. Click **OK**.
h. Save your changes to the master configuration..

If the client application uses a JMS connection factory on the server, use the WebSphere Application Server administrative console.

a. Start the WebSphere Application Server administrative console.
b. To display the default messaging provider, click **Resources -> JMS -> JMS providers**.
c. Change the **Scope** check box to set the level at which the connection factory is to be visible, according to your needs.
d. In the content pane click **Default messaging provider** to display a table of properties for the default messaging provider, including links to the types of JMS resources that it provides.
e. In the content pane, under **Additional properties**, click **Connection factories** to display any existing connection factories in the content pane.
f. Display the general properties of the connection factory.
   - To use an existing JMS connection factory, click the name of the connection factory.
   - To create a new JMS connection factory, click **New**.

   For more information about configuring a JMS connection factory, see "Configuring a unified connection factory for the default messaging provider" on page 518.
g. Ensure that the **Provider Endpoints** property includes the provider endpoint address for each bootstrap server. Type the value as a comma-separated list of endpoint addresses; for example:

   ```
   boothost1:7278,boothost2:7289:BootstrapTunneledSecureMessaging
   ```
h. Click **OK**.
i. Save your changes to the master configuration.

3. Optional: Configure the endpoint address of the bootstrap server on the Provider endpoint property of the activation specification.

   If the client application uses a JMS activation specification on the server, use the WebSphere Application Server administrative console.

   a. Start the WebSphere Application Server administrative console.
   b. To display the default messaging provider, click **Resources -> JMS -> JMS providers**.
   c. Select the default provider for which you want to configure an activation specification.
   d. Optional: Change the **Scope** check box to the scope level at which the activation specification is visible to applications, according to your needs.
   e. In the content pane, under the **Additional properties** heading, click **Activation specifications** to list any existing JMS activation specifications for the default messaging provider in the content pane.
   f. Display the properties of the JMS activation specification.
      - To use an existing JMS activation specification, click one of the names listed.

- To create a new JMS activation specification, click **New**.

For more information about configuring a JMS activation specification, see “Configuring an activation specification for the default messaging provider” on page 524.

g. Ensure that the **Provider Endpoints** property includes the provider endpoint address for each bootstrap server. Type the value as a comma-separated list of endpoint addresses; for example:

```
boothost1:7278,boothost2:7289:BootstrapTunneledSecureMessaging
```

h. Click **OK**.

i. Save your changes to the master configuration.

Protecting an MDB application from system resource problems

You can configure the system so that if there is a problem with a dependent external system resource, the enterprise application is stopped before messages are moved unnecessarily to an exception destination. This configuration also handles occasional problems with messages without blocking the enterprise application.

Before you begin

This task assumes that you have deployed an enterprise application that contains a message-driven bean (MDB) that interacts with external system resources.

The destination to which the MDB listens must use an exception destination. This exception destination can be the system default, or one configured specifically for the destination.

To complete this task, you need the following information:

- The enterprise application that contains the MDB.
- The dependent external system resources.
- An acceptable value for the **Sequential failed message threshold**, that is, the maximum number of sequential failures of delivery of messages, after which the MDB is stopped. This property applies to sets of messages.
- An acceptable value for the **Delay between failing message retries**, that is, the time in milliseconds before a failing message is available to be delivered to the MDB. Other messages might be delivered during this period, unless the**Sequential failed message threshold** and the maximum concurrency is set to 1.
- An acceptable value for the **Maximum failed deliveries per message**, that is, the maximum number of failed attempts to process a message, after which the message is forwarded from its intended destination to the exception destination. This property applies to individual messages.

About this task

When an MDB fails to process a message, the message is rolled back and made available to the MDB again. Typically, the messaging system is configured in one of the following ways:

1. Failed messages are retried a finite number of times, and if they continue to fail, they are moved to an exception destination allowing subsequent messages to be processed.
2. Failed messages are retried indefinitely until the problem is rectified.

Configuration (1) protects the MDB from an occasional problem message that prevents subsequent messages from being processed. However, if there is a prolonged problem with a resource that the enterprise application depends on, for example a database, all messages that are sent to the destination might be moved to the exception destination.

Configuration (2) blocks the delivery of messages until the original failing message problem is resolved. This configuration prevents messages being moved unnecessarily to an exception destination, but it also blocks subsequent messages as soon as a single problem message fails to be processed.

You can configure the activation specification for an MDB so that the MDB endpoint is stopped automatically when a number of failures with sequential messages are detected. These failures indicate a problem with a dependent resource. When the problem is resolved, the MDB endpoint is restarted manually. This configuration tolerates occasional message failures, allowing individual problem messages to be moved to the exception destination without blocking the entire MDB.

Use the following steps to protect an enterprise application from dependent external system resource failures.

Procedure

1. Navigate to the deployed enterprise application that contains the MDB.
2. From the MDB, navigate to its JMS activation specification. Click **Resources -> JMS -> Activation specifications -> *activation_specification_name***.
3. Enter a value for the **Sequential failed message threshold** and the **Delay between failing message retries**.
4. Save the configuration.
5. Navigate to the destination to which the MDB is listening. Click one of the following, as appropriate:
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name***
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name***
6. Enter a value for the **Maximum failed deliveries per message**.
7. Save your changes to the master configuration.

Results

You have configured the enterprise application to protect itself from the sort of external resource problem that can occur at any time. This means that, in the event of a system resource problem, the MDB is stopped automatically when the Sequential failed message threshold is reached for any message.

What to do next

When the system resource that failed becomes available, you can restart the system resource and resume the MDB. The messages that failed during the system resource downtime are retried instead of being left on an exception destination.

Example 1: Handling a planned outage of an MDB application external resource:

You can configure the system so that, if there is a problem with a dependent external system resource, the enterprise application can continue.

Before you begin

During the time that the system resource is unavailable, there must be no exceptions in the enterprise application, or messages on the exception destination that must be resolved later.

About this task

Add a maintenance level to an external system resource that is used by the deployed message-driven bean (MDB) of one of the enterprise applications. The act of applying the maintenance level requires the system resource (for example, a database) to be unavailable for about five minutes.

Procedure

1. Navigate to the deployed enterprise application that contains the MDB.

2. From the MDB, navigate to its JMS activation specification. Click **Resources -> JMS -> Activation specifications -> *activation_specification_name*** and click **Pause** on the administrative panel for the MDB.
3. When you receive a JMX notification and a log entry indicating that the MDB is paused, stop the database and apply the maintenance level. While the MDB is paused, no messages are sent to the exception destination and no error messages appear in the console related to the stopped database.
4. Restart the database and test that it is working as expected.
5. Log on to the administrative console again, navigate to the same enterprise application and click **Resume** on the administrative panel for the MDB. You can also resume the MDB by using scripting and the JCA MBean. The initial JMX notification and log entry indicate which MBean to use to resume the MDB. The MDB begins to be driven with the messages that are on the destination.

Results

You have paused and resumed an application while an external resource that it uses is not available for a short time.

Example 2: Automatically stopping an MDB when a system resource becomes unavailable:

To prepare for a system resource becoming unavailable, configure the system to stop the message-driven bean (MDB) automatically after a small number of message failures, and to alert you to the problem.

Before you begin

This task assumes that you have deployed an enterprise application containing a message-driven bean (MDB) that interacts with external system resources.

The destination to which the MDB listens must use an exception destination. This exception destination can be the system default, or one configured specifically for the destination.

To complete this task you need the following information:

- The enterprise application that contains the MDB.
- The dependent external system resource.
- A value of `3` for the **Sequential failed message threshold**. This is the maximum number of sequential failures of delivery of messages, after which the MDB is stopped. This property applies to sets of messages.
- A value of `5000` for the **Delay between failing message retries**, that is, the time in milliseconds before a failing message is available to be delivered to the MDB. Other messages might be delivered during this period, unless the**Sequential failed message threshold** and the maximum concurrency is set to `1`.
- A value of `5` for the **Maximum failed deliveries per message**, that is, the maximum number of failed attempts to process a message, after which the message is forwarded from its intended destination to the exception destination. This property applies to individual messages.

About this task

In this scenario, the enterprise application is a continuously running system that uses a deployed MDB to access an external system resource.

If the external resource experiences a problem and becomes unavailable, the deployed MDB cannot access that resource, so the transaction that is associated with the MDB is rolled back and the message, msg1, is put back on the queue.

The message msg1 is hidden for a retry delay of five seconds, set in **Delay between failing message retries**, before it is made available to the MDB.

Meanwhile, the MDB processes the next message on the queue, msg2. The external resource is still unavailable, so the processing of this message also fails. The message transaction is rolled back and the message is hidden for five seconds. The next message on the queue, msg3, is processed, fails, and is also hidden.

When the number of hidden messages reaches the **Sequential failed message threshold**, the MDB will not process any further messages until one of the hidden messages becomes available again.

When the **Delay between failing message retries** for msg1 expires, msg1 is unhidden and reprocessed. Because the resource is still unavailable, the message is rehidden. The same thing happens to msg2 and msg3.

A message is considered a failed message when it is rolled back one less than the **Maximum failed deliveries per message** limit (five times in this scenario). So after msg1 is unhidden for the fourth time, rolled back and rehidden, the sequential failure count is incremented. At this point, msg2 becomes unhidden, rolled back and rehidden. Similarly, msg3 becomes unhidden, rolled back and rehidden. The sequential failure count reaches the **Sequential failed message threshold** and the MDB stops automatically. A JMX notification is emitted by the JCA MBean and a log entry alerts the system administrator that the MDB has stopped.

Note: In this scenario, the MDB is stopped automatically when the system resource has been unavailable for approximately 20 seconds. If the system resource is unavailable for a shorter time, and the sequential failure count does not reach the **Sequential failed message threshold**, messages are processed successfully on one of the retries. In effect, the system continues normally without manual intervention, and without sending any messages to the exception destination.

Procedure

1. Navigate to the deployed enterprise application that contains the MDB.
2. From the MDB, navigate to its JMS activation specification. Click **Resources -> JMS -> Activation specifications -> *activation_specification_name***.
3. Enter a value of `3` for the **Sequential failed message threshold**.
4. Enter a value of `5000` for the **Delay between failing message retries**.
5. Save the configuration.
6. Navigate to the destination to which the MDB is listening. Click one of the following paths, as appropriate:
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name***
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name***
7. Enter a value of `5` for the **Maximum failed deliveries per message**.
8. Save your changes to the master configuration.
9. When you receive a JMX notification and a log entry indicating that the MDB (or endpoint) has been paused, investigate the problem with the system resource that the MDB was using. While the MDB is paused, no messages are sent to the exception destination and no error messages appear in the console related to the stopped database.
10. When the system resource that failed becomes available, restart it.
11. Log on to the administrative console again, navigate to the same enterprise application and click **Resume** on the administrative panel for the MDB. You can also resume the MDB by using scripting and the JCA MBean. The initial JMX notification and log entry indicate which MBean to use to resume the MDB. The MDB begins to be driven with the messages that are on the destination.

Results

You have configured the system to protect itself from external resource failures.

What to do next

When the MDB is resumed, the JCA MBean emits a JMX notification to indicate that the MDB has resumed. Messages on the queue are consumed, messages that had failed are retried, and the transaction commits.

Example 3: The system experiences problems with a problem message:

To prepare for a problem message, configure the system to move that message to an exception destination and allow other messages to be processed successfully.

Before you begin

This task assumes that you have deployed an enterprise application containing a message-driven bean (MDB) that interacts with external system resources.

The destination to which the MDB listens must use an exception destination. This exception destination can be the system default, or one configured specifically for the destination.

To complete this task, you need the following information:

- The enterprise application that contains the MDB.
- The dependent external system resources.
- Set a value of `3` for the **Sequential failed message threshold**. This is the maximum number of sequential failures of delivery of messages, after which the MDB is stopped. This property applies to sets of messages.
- Set a value of `5000` for the **Delay between failing message retries**, that is, the time in milliseconds before a failing message is available to be delivered to the MDB. Other messages might be delivered during this period, unless the**Sequential failed message threshold** and the maximum concurrency is set to `1`.
- Set a value of `5` for the **Maximum failed deliveries per message**, that is, the maximum number of failed attempts to process a message, after which the message is forwarded from its intended destination to the exception destination. This property applies to individual messages.

About this task

In this scenario, the enterprise application is a continuously running system that uses a deployed MDB to access an external system resource.

When a problem message (msg1 in this scenario) is encountered, it is put back on the queue.

Instead of msg1 being made available to the MDB immediately, it is hidden for the **Delay between failing message retries** retry delay of five seconds.

The next message on the queue (msg2) is processed by the MDB. This message and subsequent messages succeeds.

When the **Delay between failing message retries** for msg1 expires, msg1 is unhidden and reprocessed. It is put back on the queue again.

The MDB continues to process subsequent messages normally but each time msg1 is processed, it is put back on the queue.

When the number of times msg1 has been unhidden, rolled back and rehidden reaches the **Maximum failed deliveries per message** limit (five times in this scenario), it is moved to the configured exception destination.

Procedure

1. Navigate to the deployed enterprise application that contains the MDB.
2. From the MDB, navigate to its JMS activation specification. Click **Resources -> JMS -> Activation specifications -> *activation_specification_name***.
3. Enter a value of `3` for the **Sequential failed message threshold**.
4. Enter a value of `5000` for the **Delay between failing message retries**.
5. Save the configuration.
6. Navigate to the destination to which the MDB is listening. Click one of the following paths, as appropriate:
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name***
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name***
7. Enter a value of `5` for the **Maximum failed deliveries per message**.
8. Save your changes to the master configuration..

Results

You have configured the system to protect itself from external resource failures and send problem messages to the exception destination.

Example 4: Automatically stopping an MDB when no exception destination is specified:

To prepare for a system resource becoming unavailable or a problem message, configure the system to stop the message-driven bean (MDB) automatically. To maintain message ordering, do not use an exception destination.

Before you begin

This task assumes that you have deployed an enterprise application containing a message-driven bean (MDB) that interacts with external system resources.

The destination to which the MDB listens must not use an exception destination, that is, the exception destination for the queue or topic space destination must be configured as none.

To complete this task, you need the following information:

- The enterprise application that contains the MDB.
- The dependent external system resources.
- A value of `1` for the **Sequential failed message threshold**. This is the maximum number of sequential failures of delivery of messages, after which the MDB is stopped. This property applies to sets of messages.

  Note: If this property is set to a value greater than 1, the system automatically resets it to 1 when an exception destination is configured as none.
- A value of `5000` for the **Delay between failing message retries**, that is, the time in milliseconds before a failing message is available to be delivered to the MDB. Other messages might be delivered during this period, unless the**Sequential failed message threshold** and the maximum concurrency is set to `1`.

- An acceptable value for the **Maximum failed deliveries per message**, that is, the maximum number of failed attempts to process a message. This property applies to individual messages.

About this task

In this scenario, the enterprise application is a continuously running system that uses a deployed MDB to access an external system resource.

When a problem message (msg1 in this scenario) is encountered, it is put back on the queue.

Instead of msg1 being made available to the MDB immediately, it is hidden for the **Delay between failing message retries** retry delay (5 seconds in this scenario).

When the number of hidden messages reaches the **Sequential failed message threshold**, the MDB will not process any further messages until one of the hidden messages becomes re-available. In this scenario, this threshold is reached as soon as msg1 is hidden.

When the **Delay between failing message retries** for msg1 expires, msg1 is unhidden and reprocessed.

This process is repeated until msg1 reaches its **Maximum failed deliveries per message** limit (five times in this scenario).

After msg1 is unhidden for the fourth time, rolled back and rehidden, the **Sequential failed message threshold** is reached and the MDB stops automatically. A JMX notification is emitted by the JCA MBean and a log entry alerts the system administrator that the MDB has stopped.

Procedure

1. Navigate to the deployed enterprise application that contains the MDB.
2. From the MDB, navigate to its JMS activation specification. Click **Resources -> JMS -> Activation specifications -> *activation_specification_name***.
3. Enter a value of `1` for the **Sequential failed message threshold**.
4. Enter a value of `5000` for the **Delay between failing message retries**.
5. Save the configuration.
6. Navigate to the destination to which the MDB is listening. Click one of the following paths, as appropriate:
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name***
   - **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name***
7. Under **Exception destination**, select **None**.
8. Enter a value of `5` in **Maximum failed deliveries per message**.
9. Save your changes to the master configuration.
10. When you receive a JMX notification and a log entry indicating that the MDB (or endpoint) has been paused, investigate the problem with the system resource that the MDB was using. While the MDB is paused, because no exception destination is configured, msg1 remains on the queue. No other messages are processed.
11. If you resume the MDB but the problem with the failing message continues, the maximum failed deliveries limit is reached on the first retry of the message, but because no exception destination is configured, the message is not moved to another queue. Instead, the whole queue point is blocked to all consumers for the Delay between failing message retries retry delay interval (5 seconds in this scenario). After this time, consumers begin again. If the failing message is still there, and fails again, the queue point is blocked for another 5 seconds. This process continues until you remove the failing

message from the queue, either by deleting it manually or solving the problem with it, and in doing so allowing the consuming application to succeed in processing.

12. Log on to the administrative console again, navigate to the same enterprise application and click **Resume** on the administrative panel for the MDB. You can also resume the MDB by using scripting and the JCA MBean. The initial JMX notification and log entry indicate which MBean to use to resume the MDB. The MDB begins to be driven with the messages that are on the destination.

Results

You have configured the system to protect itself from external resource failures while maintaining message ordering.

What to do next

When the MDB is resumed, the JCA MBean emits a JMX notification to indicate that the MDB has resumed. Messages on the queue are consumed, messages that had failed are retried, and the transaction commits.

Sample JMS 1.1 application client

If you are new to JMS 1.1 application client programming, you can use this example code as a starting-point for developing your client application.

Example

Here is a typical example of JMS 1.1 application client code:

```
import java.util.Hashtable;
import javax.jms.JMSException;
import javax.naming.Context;
import javax.naming.*;
import javax.jms.*;


public class JMSppSampleClient
{
  public static void main(String[] args)
  throws JMSException, Exception

  {
    String  messageID                = null;
    String  outString                = null;
    String  cfName                   = "jms/blueconfactory";
    String  qnameIn                  = "java:comp/env/jms/Q1";
    String  qnameOut                 = "jms/bluequename";
    boolean verbose                  = false;

    Session               session    = null;
    Connection            connection = null;
    ConnectionFactory     cf         = null;
    MessageProducer       mp         = null;
    Destination           destination = null;

    try {

     Hashtable env = new Hashtable();
     env.put(Context.INITIAL_CONTEXT_FACTORY,
          "com.ibm.websphere.naming.WsnInitialContextFactory");
     env.put(Context.PROVIDER_URL, "iiop://localhost:2809");
     Context initialContext = new InitialContext(env);
  System.out.println("Getting Connection Factory");
```

```
cf= (ConnectionFactory)initialContext.lookup( cfName );

System.out.println("Getting Queue");
destination =(Destination)initialContext.lookup(qnameOut);

      System.out.println("Getting Connection for Queue");
connection = cf.createConnection();

      System.out.println("staring the connection");
connection.start();

      System.out.println("creating session");
session = connection.createSession(false, 1);

      System.out.println("creating messageProducer");
mp = session.createProducer(destination);

      System.out.println("creating TextMessage");
TextMessage outMessage = session.createTextMessage("this is test application");

      System.out.println("sending Message");
mp.send(outMessage);

mp.close();
session.close();
connection.close();
  }
  catch (Exception je)     {}
```

Interoperating with a WebSphere MQ network

The default messaging provider (service integration) can interoperate with a WebSphere MQ network by using a WebSphere MQ link or a WebSphere MQ server. Alternatively, you can use WebSphere MQ as your messaging provider. Each type of connectivity is designed for different situations, and provides different advantages. Choose the most appropriate interoperation method for each of your messaging applications.

About this task

WebSphere Application Server can interoperate with WebSphere MQ in the following ways:

- By configuring WebSphere MQ as an external JMS provider by using the WebSphere MQ messaging provider.
- By connecting a service integration bus to a WebSphere MQ network by using the default messaging provider and a WebSphere MQ link.
- By integrating WebSphere MQ queues into a bus by using the default messaging provider and a WebSphere MQ server.

A WebSphere MQ link provides a traditional WebSphere MQ -style solution to connecting resources. A WebSphere MQ server adds the ability to directly access WebSphere MQ queues from within a bus.

Table 48. The different ways of interoperating with WebSphere MQ. Table 1 has three columns. Each column contains a figure showing a different way of interoperating with WebSphere MQ. The left column describes interoperation using the WebSphere MQ messaging provider, with no bus. The middle column describes interoperation using a WebSphere MQ network as a foreign bus. The right column describes interoperation using a WebSphere MQ queue manager or queue-sharing group as a bus member.

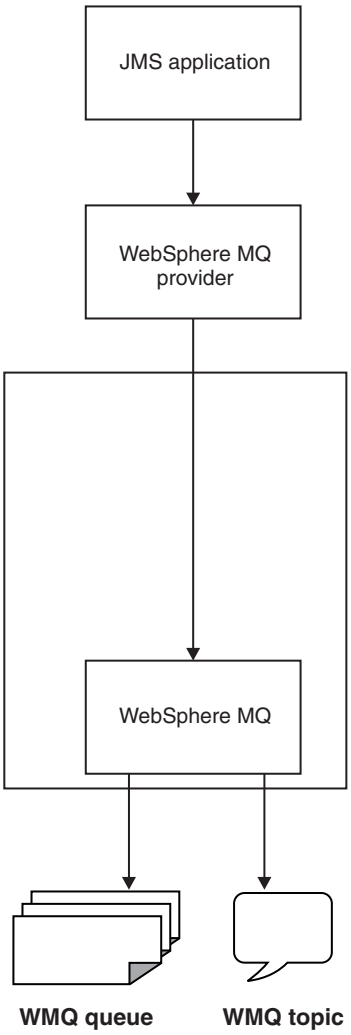
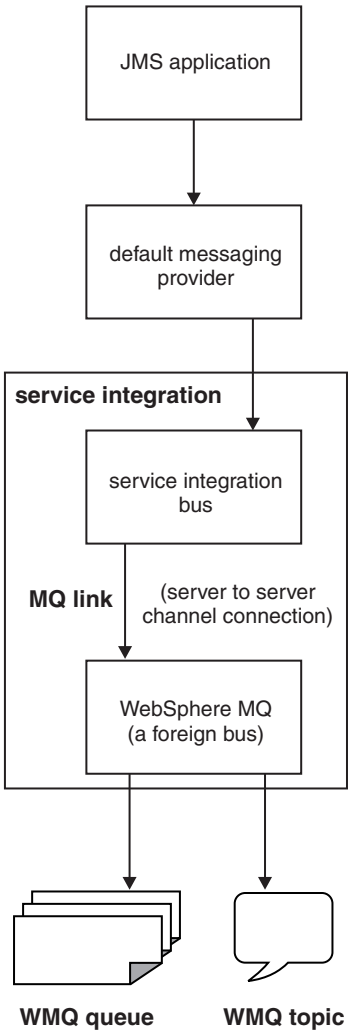
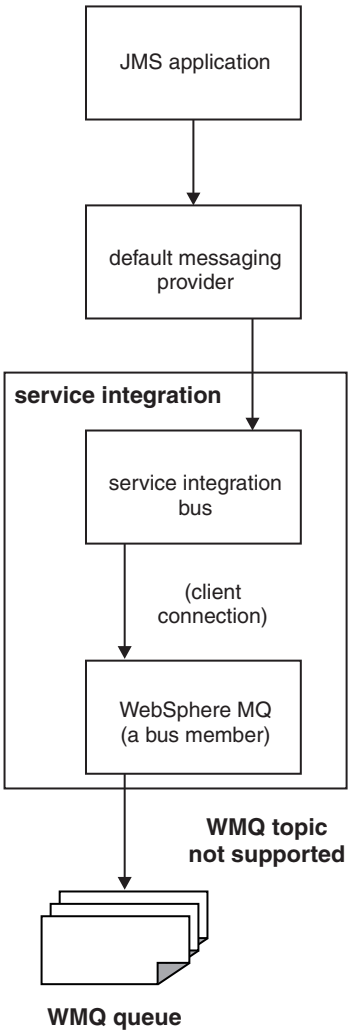
WebSphere MQ messaging provider (no bus)	**WebSphere MQ network as a foreign bus (using a WebSphere MQ link)**	**WebSphere MQ queue manager or queue-sharing group as a bus member (using a WebSphere MQ server)**
		
In this figure, a JMS application uses APIs to send a message to WebSphere MQ, for a topic or queue, via the WebSphere MQ messaging provider.	In this figure a JMS application uses the default messaging provider to pass a message to a local service integration bus. The local bus passes the message to a foreign bus, which forwards it across a WebSphere MQ link to a WebSphere MQ queue manager or queue sharing group that acts as the gateway to the WebSphere MQ network. Service integration views the WebSphere MQ network as if it were a foreign bus.	In this figure, a JMS application uses the default messaging provider to pass a message to a service integration bus. The bus passes the message through a WebSphere MQ server direct to a WebSphere MQ queue. Service integration views the WebSphere MQ server (a WebSphere MQ queue manager or queue-sharing group, and its associated queues) as a member of the local bus.

Table 48. The different ways of interoperating with WebSphere MQ (continued). Table 1 has three columns. Each column contains a figure showing a different way of interoperating with WebSphere MQ. The left column describes interoperation using the WebSphere MQ messaging provider, with no bus. The middle column describes interoperation using a WebSphere MQ network as a foreign bus. The right column describes interoperation using a WebSphere MQ queue manager or queue-sharing group as a bus member.

WebSphere MQ messaging provider (no bus)	WebSphere MQ network as a foreign bus (using a WebSphere MQ link)	WebSphere MQ queue manager or queue-sharing group as a bus member (using a WebSphere MQ server)
The WebSphere MQ messaging provider does not use service integration. It provides JMS messaging access to WebSphere MQ from WebSphere Application Server.	A WebSphere MQ link provides a server to server channel connection between a service integration bus and a WebSphere MQ queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network.When you use a WebSphere MQ link, the messaging bus is seen by the WebSphere MQ network as a virtual queue manager, and the WebSphere MQ network is seen by service integration as a foreign bus. A WebSphere MQ link allows WebSphere Application Server applications to send point-to-point messages to WebSphere MQ queues (defined as destinations in the service integration bus), and allows WebSphere MQ applications to send point-to-point messages to destinations in the service integration bus (defined as remote queues in WebSphere MQ). The link also allows WebSphere Application Server applications to subscribe to messages published by WebSphere MQ applications, and WebSphere MQ applications to subscribe to messages published by WebSphere Application Server applications. The link ensures that messages are converted between the formats used by WebSphere Application Server and those used by WebSphere MQ.	A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. For interoperation with WebSphere Application Server Version 7.0 or later, the version of WebSphere MQ must be WebSphere MQ for z/OS Version 6 or later, or WebSphere MQ (distributed platforms) Version 7 or later. A WebSphere MQ server supports the high availability and optimum load-balancing characteristics provided by a WebSphere MQ for z/OS network. A WebSphere MQ server defines the connection and quality of service properties used for the connection, and also ensures that messages are converted between the formats used by WebSphere Application Server and those used by WebSphere MQ.

For more information about these approaches, see Interoperation with WebSphere MQ.

To interoperate with a WebSphere MQ network complete one or more of the following steps.

Procedure

- Choose the most appropriate interoperation method for each of your messaging applications. Complete this step if your existing or planned messaging environment involves both WebSphere MQ and WebSphere Application Server systems, and it is not clear to you whether you should use the default messaging provider, the WebSphere MQ messaging provider, or a mixture of the two.
- Configure the WebSphere MQ messaging provider.
- Use WebSphere MQ links to connect a bus to a WebSphere MQ network.
- Use WebSphere MQ server to integrate WebSphere MQ queues into a bus.

Using WebSphere MQ links to connect a bus to a WebSphere MQ network

If you operate within a WebSphere Application Server environment, sending messages across a service integration bus, you can also exchange point-to-point and publish/subscribe messages with applications in a WebSphere MQ network. To do this, you configure a foreign bus connection that links to a WebSphere MQ network through a WebSphere MQ link.

Before you begin

Decide which method to use to configure these resources. You can configure a WebSphere MQ link by using the administrative console as described in this task, or you can configure a WebSphere MQ link by using the wsadmin tool.

About this task

A WebSphere MQ link provides a server to server channel connection between a service integration bus and a WebSphere MQ queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network. The link operates on a messaging engine in a service integration bus to provide functions that simplify and automate interoperation with WebSphere MQ.

Using the WebSphere MQ link panels of the WebSphere Application Server administration console, you can choose:

- The WebSphere MQ queue manager or queue-sharing group in the WebSphere MQ network through which your messages will flow
- Whether to enable WebSphere Application Server applications to publish and subscribe to a message broker in the WebSphere MQ network

Procedure

- Learn about interoperating with a WebSphere MQ network.
- Create a new WebSphere MQ link.
- Administer an existing WebSphere MQ link.
- Create applications that can interoperate with WebSphere MQ.

Creating a new WebSphere MQ link:

A WebSphere MQ link provides a server to server channel connection between a service integration bus and a WebSphere MQ queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network. Use the foreign bus connection wizard to create a foreign bus and link it to the WebSphere MQ network through a WebSphere MQ link.

Before you begin

Decide which method to use to configure these resources. You can create a new WebSphere MQ link by using the administrative console as described in this task, or you can create a new WebSphere MQ link by using the wsadmin tool.

The following resources must be defined in WebSphere Application Server:

- A service integration bus that you want to connect from (known as the local bus) with at least one bus member.

The following resources must be defined in WebSphere MQ:

- A queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network.
- A listener that is configured and running.
- (optionally) A sender channel to receive messages on the local bus, a receiver channel to send messages from the local bus, or both.
- (For publish-subscribe messaging) A topic and input queue for broker publish-subscribe flow.

About this task

You use the foreign bus connection wizard to connect a bus and a WebSphere MQ network. You can configure the connection for either point-to-point or publish-subscribe messaging.

Specifically, the wizard helps you configure the following resources:

- The bus and messaging engine on which the WebSphere MQ link is defined.
- The foreign bus that represents the WebSphere MQ network.
- The WebSphere MQ link.
- (Optionally) The sender and receiver channels and protocol. Using the wizard you can choose to define no channels (and add them afterward), or one channel if your WebSphere MQ link is to be one-way, or both channels.
- (Optionally) Security for messages flowing across the link.
- (Optionally) A publish/subscribe broker profile and associated topic mappings, to allow publication and subscription with a broker in the WebSphere MQ network.

The wizard does not ask you to set all possible properties of a WebSphere MQ link, and many of the properties are set to default values. You can fine tune these properties afterward if necessary, by modifying the WebSphere MQ link.

For a sample configuration showing a systems view of the setup for a WebSphere MQ link, see "WebSphere MQ link sample configuration" on page 550.

Procedure

1. Use the foreign bus connection wizard to connect a bus and a WebSphere MQ network to use point-to-point messaging or publish-subscribe messaging.

   A WebSphere MQ link is created and configured as part of the action of the wizard.

   Note: You can subsequently convert a point-to-point connection to a publish/subscribe connection, by adding a publish/subscribe broker on the WebSphere MQ link for the connection.
2. Optional: Modify the new WebSphere MQ link.

   When you create a new WebSphere MQ link, the following properties are set to default values:

   - Description
   - Adoptable
   - Exception destination
   - Initial state
   - Nonpersistent message speed

   You can fine tune these properties by modifying the link.
3. Optional: Add or modify the WebSphere MQ receiver channel.

   If you did not choose to **Enable Service integration bus to WebSphere MQ message flow** in the foreign bus connection wizard, you have not yet defined a WebSphere MQ receiver channel. If you did choose this option in the wizard, you have defined the receiver channel name, host name and communication port, and the wizard has used default values for the following properties:

   - Inbound nonpersistent message reliability
   - Inbound persistent message reliability
   - Prefer queue points local to this link's messaging engine
   - Initial state

   You can fine tune these properties by modifying the channel.
4. Optional: Add or modify the WebSphere MQ sender channel

If you did not choose to **Enable WebSphere MQ to Service integration bus message flow** in the foreign bus connection wizard, you have not yet defined a WebSphere MQ sender channel. If you did choose this option in the wizard, you have defined the sender channel name, host name, communication port and transport chain, and the wizard has used default values for the following properties:

- Disconnect interval
- Short retry count
- Short retry interval
- Long retry count
- Long retry interval
- Initial state

You can fine tune these properties by modifying the channel.

WebSphere MQ link sample configuration:

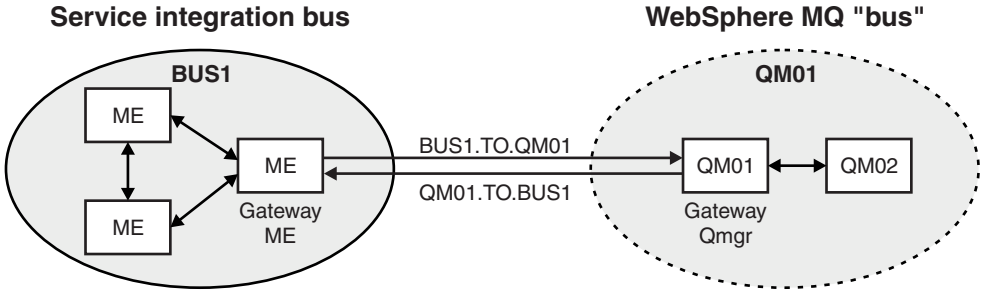
This sample configuration illustrates how you can use a WebSphere MQ link to connect WebSphere Application Server service integration messaging to WebSphere MQ messaging.

A WebSphere MQ link connects one service integration messaging engine, called the gateway messaging engine, to one WebSphere MQ queue manager or queue-sharing group, called the gateway queue manager. All messaging engines in the service integration bus use the gateway messaging engine to route messages to and from the WebSphere MQ network; all queue managers and queue-sharing groups in the WebSphere MQ network use the gateway queue manager to route messages to and from the service integration bus.

Typically, a WebSphere MQ link consists of two TCP/IP connections:

- The WebSphere MQ link sender channel, which carries messages from service integration to WebSphere MQ.
- The WebSphere MQ link receiver channel, which carries messages from WebSphere MQ to service integration.

WebSphere MQ calls these TCP/IP connections message channels, a receiver channel which connects to the WebSphere MQ link sender channel and a sender channel which connects to the WebSphere MQ link receiver channel. The following figure shows a configuration like the one just described, with a WebSphere MQ link sender channel called BUS1.TO.QM01 and a WebSphere MQ link receiver channel called QM01.TO.BUS1.



If you only require messages to flow in one direction, you need only define one TCP/IP connection. For example, a WebSphere MQ link sender channel in service integration that connects to a receiver channel in WebSphere MQ is enough to support message flow from service integration to WebSphere MQ. However, this sample builds a configuration that allows messages to flow in both directions.

Sample configuration context

The purpose of this sample is to connect a WebSphere Application Server configuration to a WebSphere MQ configuration so that asynchronous messages can flow in both directions between the two messaging systems. The sample assumes that you have already set up a WebSphere Application Server configuration like this:

- An application server called server1 located on a node called London. In a Network Deployment, server1 might be one of several servers in a cell and might be one of several servers in a cluster, but this sample is equally applicable to a base deployment containing just one application server.
- The IP host name for the server London is LONDON.
- A service integration bus called BUS1.
- server1 is a member of BUS1; the messaging engine it contains is called London.server1-BUS1.
- A queue-type bus destination called ServiceIntegrationQueue1, which is one of the destinations in BUS1.

The sample also assumes that you already have a WebSphere MQ configuration like this:

- Queue managers called QM01 and QM02 which are part of a network of interconnected WebSphere MQ queue managers and queue-sharing groups. If you have only one queue manager then you can ignore references to QM02 in this sample.
- The IP host name for the server where QM01 runs is PARIS.
- A queue called WMQ11 which is located on QM01 and a queue called WMQ21 which is located on QM02. There might be many other queues defined in the WebSphere MQ network but this sample is concerned only with the two WebSphere MQ queues that you are going to access from WebSphere Application Server.

You select London.server1-BUS1 to be the gateway messaging engine and QM01 to be the gateway queue manager.

Sample configuration for the connections

This section describes the settings that you or your WebSphere MQ administrator need to configure for the connections:

- The commands that your WebSphere MQ administrator uses to configure the WebSphere MQ components that correspond to the WebSphere MQ link:
  - The sender channel
  - The receiver channel
  - The transmission queue

  For JMS programs, the WebSphere MQ administrator also defines a JMS destination that identifies the queue in the service integration bus. Refer to the WebSphere MQ documentation for more details about these commands.
- The parameters that you need when you use the WebSphere Application Server administrative console to configure:
  - A foreign bus connection, which includes the foreign bus representing the network of WebSphere MQ queue managers and queue-sharing groups, and the WebSphere MQ link representing the connection to that network
  - JMS destinations that identify queues in the WebSphere MQ network

After you configure and activate these components your applications can exchange messages between WebSphere Application Server service integration messaging and WebSphere MQ messaging. Optionally you can configure additional administrative artifacts that allow you more detailed control over the queues and destinations, see “Sample configuration for the destinations” on page 553.

WebSphere MQ command to configure the sender channel

```
DEFINE    CHL(QM01.TO.BUS1) +
          CHLTYPE(SDR) +
          TRPTYPE(TCP) +
          CONNAME('LONDON(5558)') +
          XMITQ(BUS1)
```

Your WebSphere MQ administrator chooses the name for the sender channel, which in this sample is QM01.TO.BUS1.

The CONNAME parameter specifies the IP host and port of the gateway messaging engine.

The XMITQ parameter specifies the name of the transmission queue, which is normally the same as the virtual queue manager name of the service integration bus, which is preferably the same as the bus name.

WebSphere MQ command to configure the receiver channel

```
DEFINE    CHL(BUS1.TO.QM01) +
          CHLTYPE(RCVR) +
          TRPTYPE(TCP)
```

Your WebSphere MQ administrator chooses the name for the receiver channel, which in this sample is BUS1.TO.QM01.

WebSphere MQ command to configure the transmission queue

```
DEFINE   QL(BUS1) +
         USAGE(XMITQ)
```

Your WebSphere MQ administrator chooses the name for the transmission queue, but it is convenient to use the name of the service integration bus BUS1. If the service integration bus name is not a valid WebSphere MQ queue manager name then the WebSphere Application Server administrator must define a different virtual queue manager name for use here.

WebSphere MQ JMSAdmin command to configure the JMS destination

```
DEFINE   Q(ServiceIntegrationQueue1) +
         QMGR(BUS1) +
         QUEUE(ServiceIntegrationQueue1)
```

Your WebSphere MQ JMS applications can use this JMS destination to send messages to the service integration bus destination ServiceIntegrationQueue1 in BUS1.

WebSphere Application Server parameters for the foreign bus connection

You configure a foreign bus connection as part of the topology of the service integration bus. For this sample, the service integration bus is BUS1 and the foreign bus connection uses the following settings:

Bus connection type	Direct connection
Foreign bus type	WebSphere MQ
Messaging engine to host the connection	London.server1-BUS1
Virtual queue manager name	BUS1 (use the name of the local bus)
Foreign bus name	QM01 (use the name of the WebSphere MQ gateway queue manager)
MQ link name	TO.QM01
Enable Service integration bus to WebSphere MQ message flow	Checked (default)
WebSphere MQ receiver channel name	BUS1.TO.QM01
Host name	PARIS
Port	1414
Enable WebSphere MQ to Service integration bus message flow	Checked (default)
WebSphere MQ sender channel name	QM01.TO.BUS01

There are other options in the wizard relating to publish/subscribe messaging and security. Leave these settings to default.

WebSphere Application Server parameters for JMS destinations

You configure JMS destinations to allow service integration JMS applications to access queues in the WebSphere MQ network. This sample needs JMS destinations for queue WMQ11 on queue manager QM01, and for queue WMQ21 on queue manager QM02.

Note that these JMS destinations are WebSphere MQ queues but for the purposes of this sample you are accessing these queues from service integration JMS programs so you need to define JMS destinations for the default messaging provider (service integration) not for the WebSphere MQ JMS provider.

For WMQ11, configure the following parameters:

Name	WMQ11
JNDI name	jms/WMQ11
Bus name	QM01
Queue name	WMQ11

Leave all other settings to default.

For WMQ21, configure the following parameters:

Name	WMQ21
JNDI name	jms/WMQ21
Bus name	QM01
Queue name	WMQ21@QM02

Leave all other settings to default.

Sample configuration for the destinations

The sample JMS destinations in Sample configuration for the connections point directly to the corresponding WebSphere MQ queues and service integration destinations. If you prefer, you can configure additional components so that:

- The WebSphere MQ JMS destination points to a WebSphere MQ queue (actually a remote or alias queue) which points to the service integration destination.
- The service integration JMS destinations point to service integration destinations (actually foreign or alias destinations) which point to the WebSphere MQ queues.

Refer to the WebSphere MQ documentation for information about when and how to define remote and alias queues.

WebSphere Application Server parameters for foreign destinations

You configure foreign destinations for WebSphere MQ queues to allow control over how service integration applications access each queue. For example, you can configure foreign destinations for each of two queues and specify that service integration includes an MQRFH2 header in messages to one queue but not to the other.

You configure foreign destinations as destination resources of the service integration bus.

For WMQ11, configure the following parameters:

Identifier	WMQ11
Bus	QM01

Leave all other settings to default.

For WMQ21, configure the following parameters:

Identifier	WMQ21@QM02
Bus	QM01

Leave all other settings to default.

After you define these foreign destinations you can, for example, set the _MQRFHAllowed custom property for either destination or both, as required.

Administering an existing WebSphere MQ link:

The WebSphere MQ link enables the exchange of point-to-point and publish/subscribe messages with a WebSphere MQ network. After you have created a WebSphere MQ link you can undertake various administrative actions on the link.

About this task

The WebSphere MQ link connects a WebSphere Application Server and a WebSphere MQ network. Because these two systems are working together, their functions must be in step so that each is aware of the status of the other. Administrative actions on the WebSphere MQ link and its functions include the following:

Procedure

- Modify individual functions of the WebSphere MQ link .

  You can modify the link itself, add or modify the sender channel and receiver channel, or define a broker profile and associated topic mappings.
- Modify security for a WebSphere MQ link.
- View the status of a WebSphere MQ link and its components. You can view the status of a WebSphere MQ link and its sender and receiver channels, and you can view the status of subscriptions for a WebSphere MQ link publish/subscribe broker profile.
- Start a WebSphere MQ link.
- Stop a WebSphere MQ link.

  When you stop the link, all its functions are stopped too. For example, stopping a WebSphere MQ link with broker profiles on it might leave a message broker in the WebSphere MQ network with a backlog of messages. For more information, see “Stopping a WebSphere MQ link” on page 568, “Stopping the sender channel on a WebSphere MQ link” on page 570 and “Stopping the receiver channel on a WebSphere MQ link” on page 569.
- Delete a WebSphere MQ link or one of its components. When you remove a foreign bus connection between a service integration bus and a WebSphere MQ network, you also delete the associated WebSphere MQ link along with any publish/subscribe broker profiles and topic mappings. You can also delete a WebSphere MQ link publish/subscribe broker profile or delete a topic mapping from a WebSphere MQ link.

Modifying a WebSphere MQ link:

How and when to modify the properties of a WebSphere MQ link.

Before you begin

Decide which method to use to configure these resources. You can modify a WebSphere MQ link by using the administrative console as described in this task, or you can modify a WebSphere MQ link by using the wsadmin tool.

You have to know the name of the bus, and the messaging engine on the bus that contains the WebSphere MQ link you want to modify.

About this task

A WebSphere MQ link provides a server to server channel connection between a service integration bus and a WebSphere MQ queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network. At any time after you create a new WebSphere MQ link, you can modify its properties.

When you use the foreign bus connection wizard to connect a bus and a WebSphere MQ queue manager or queue-sharing group (known as the "gateway queue manager") to use point-to-point messaging or publish-subscribe messaging, one or more WebSphere MQ links are created and configured as part of the task. However the wizard does not ask you to set all possible properties of a WebSphere MQ link, and some of the link properties are not set or are set to default values. You can fine tune these properties by modifying the link.

When you create a new WebSphere MQ link, you can also choose not to create sender or receiver channels. These channels can be added later by modifying the link.

When you use the foreign bus connection wizard to create a WebSphere MQ link for point-to-point messaging, you can later modify the link for publish-subscribe messaging by adding a publish/subscribe broker profile to the link.

Procedure

1. In the navigation pane, click, **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name***.
2. Modify the properties of the link.

   For information about all the properties that you can modify, see "WebSphere MQ link [Settings]" on page 2228.

   When you create a new WebSphere MQ link, the following properties are set to default values:

   Description
   An optional description for the WebSphere MQ link, for administrative purposes.

   Adoptable
   Whether or not a running instance of a WebSphere MQ link receiver channel (associated with this MQ link) should be adopted or not. In the event of a communications failure, it is possible for a running instance of a WebSphere MQ link receiver channel to be left waiting for messages. When communication is re-established, and the partner WebSphere MQ sender channel next attempts to establish a session with the WebSphere MQ link receiver channel, the request will fail as there is already a running instance of the WebSphere MQ link receiver channel that believes it is in session with the partner WebSphere MQ sender channel. You can overcome this problem by selecting this option, which causes the already running instance of the WebSphere MQ link receiver channel to be stopped and a new instance to be started. By default, this option is not selected.

   Exception destination
   The destination for an inbound message when the WebSphere MQ link cannot deliver the message to its target bus destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist. By default, undeliverable messages are sent to the system default exception destination for the messaging engine that this link is assigned to. However, it can aid problem-solving if you separate these exceptions out from other system messages by configuring a specific exception destination for this link.

   Initial state
   Whether the WebSphere MQ link is started or stopped when the hosting messaging engine is first started. Until started, the WebSphere MQ link is unavailable. By default, the value is "Started".

Nonpersistent message speed
The class of service for nonpersistent messages on channels of this WebSphere MQ link. By default, the value is “Fast”.

3. Configure the additional properties. You can configure any of the following additional properties of this WebSphere MQ link:
   - Publish/subscribe broker profiles
   - Receiver channel
   - Sender channel
   - Sender channel transmitters
4. Configure the related items. You can configure any of the following related items of this WebSphere MQ link:
   - Foreign bus connection
   - Link transmitters
5. Save your changes to the master configuration.
6. If you have enabled dynamic configuration updates, the changes take effect immediately (or on channel restart if you also modified WebSphere MQ link sender or receiver channels), otherwise restart the application server.

Adding or modifying a publish/subscribe broker on the WebSphere MQ link:

A publish/subscribe broker profile, and associated topic mappings, allows publication and subscription with a broker in a WebSphere MQ network. You can use the administrative console to define a broker profile on a WebSphere MQ link, forming a publish/subscribe bridge with a WebSphere MQ network.

Before you begin

You have to know the name of the bus, messaging engine name, and the name of the WebSphere MQ link on which you intend to create or modify the broker profile. You also have to know the queue manager name for the message broker in the WebSphere MQ network where the input queues for the required publication message flows are located.

About this task

When you use the foreign bus connection wizard to connect a bus and a WebSphere MQ queue manager to use publish/subscribe messaging, you can define a publish/subscribe broker profile and associated topic mappings. Alternatively, you can use the foreign bus connection wizard to connect a bus and a WebSphere MQ queue manager to use point-to-point messaging, then later modify the WebSphere MQ link for publish/subscribe messaging by adding a publish/subscribe broker profile and associated topic mappings to the link.

After you have created the broker profile you must ensure that the service integration bus has sufficient authority on the message broker instance to send subscription requests.

Procedure

1. In the navigation pane, click one of the following paths:
   - **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles**
   - **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles**
2. In the content pane, either click **New** to add a new broker profile, or click the name of a existing broker profile that you want to modify.

3. Add or modify the properties of the broker profile.

   For information about the broker profile properties (name, description, broker queue manager name), see “Publish/subscribe broker profiles [Settings]” on page 2139.

   For an existing broker profile you can only modify the description. Note that, for an existing broker profile created by using the foreign bus connection wizard, the broker profile name was generated automatically by adding “_broker_profile” to the end of the broker queue manager name.
4. Optional: Under Additional properties, configure the topic mappings for this broker profile. For more information, see “Adding or modifying topic mappings on the WebSphere MQ link publish/subscribe broker” on page 558.
5. Click **OK**.
6. Save your changes to the master configuration.

What to do next

After you have created the broker profile you must ensure that the service integration bus has sufficient authority on the message broker instance to send subscription requests. You can do this either by modifying the message broker configuration on the WebSphere MQ network, or by modifying the service integration bus configuration. See “Defining permissions for a WebSphere MQ link publish/subscribe broker to work with WebSphere MQ.”

Defining permissions for a WebSphere MQ link publish/subscribe broker to work with WebSphere MQ:

There are several ways to ensure that a service integration bus has authority with a message broker (in a WebSphere MQ network) to send subscription requests.

Before you begin

Before you start this task, you must have created a broker profile, part of a publish/subscribe bridge on a WebSphere MQ link.

About this task

When you have created a broker profile you must ensure that the service integration bus has sufficient authority on the message broker instance to send subscription requests. You can do this either by:

- Modifying the message broker configuration in the WebSphere MQ network if you are a WebSphere MQ administrator. Or
- Modifying the service integration bus configuration if you are a WebSphere Application Server administrator.

To ensure authority, complete one of the following steps:

Procedure

1. Modify the message broker configuration by granting WebSphere MQ permissions for the user "SIBServer", which is the user name under which the publish/subscribe bridge control messages will be published.
2. Modify the service integration bus by setting the OutboundUserID of the foreign bus that represents the WebSphere MQ network (which you defined as part of the WebSphere MQ link configuration). The OutboundUserID should be set to a user ID that already has the relevant WebSphere MQ permissions on the message broker installation image. By doing this you can be sure the publish/subscribe bridge control messages will arrive at the message broker in the WebSphere MQ network with the appropriate user ID set in them.

What to do next

You are now ready to define topic mappings on the publish/subscribe broker profile.

Adding or modifying topic mappings on the WebSphere MQ link publish/subscribe broker:

Define topic mappings on a publish/subscribe broker profile, part of the publish/subscribe bridge on a WebSphere MQ link. A topic mapping is a mapping between a topic on a service integration bus and a stream queue and subscription point provided by a WebSphere MQ broker.

Before you begin

You have to know the bus name, messaging engine name, WebSphere MQ link name and the name of the broker profile on which you intend to define the topic mappings.

About this task

Topic mappings are associated with a WebSphere MQ link publish/subscribe broker profile. Together, the profile and topic mappings enable publication and subscription with a broker in a WebSphere MQ network.

Note: Publication messages forwarded to a message broker in the WebSphere MQ network are republished on the same topic as they were originally published to in the service integration bus topic space, and vice versa.

To define topic mappings, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name* -> [Additional Properties] Topic mappings**
2. In the content pane, either click **New** to add a new topic mapping, or click the name of a existing topic mapping that you want to modify. The “Topic Mapping [Settings]” on page 2173 form is displayed.
3. Type the topic name.

   The name of the topic on the service integration bus. The name must be the same as the topic name on the message broker in a WebSphere MQ network.

   The topic name can contain wild cards that are in the service integration bus syntax. For more information, see Wild cards in topic mapping.
4. Select the name of the topic space that contains the topic. If you don't select a name the default is used.
5. Select the direction of publication flow. The direction of publication flow can be:

   Bi-directional
   Messages flow in both directions between the bus and WebSphere MQ.

   To WebSphere MQ
   Messages flow only from the bus to WebSphere MQ. That is, from WebSphere Application Server to a message broker in the WebSphere MQ network.

   From WebSphere MQ
   Messages flow only to the bus from WebSphere MQ. That is, from a message broker in the WebSphere MQ network to WebSphere Application Server.
6. Select the broker stream queue you want to use. If the queue you require is not on the list, click **other, please specify** then enter the name of the broker stream queue.

The broker stream queue in this instance is a queue on the WebSphere MQ queue manager to which the message broker is connected. This queue is being used as the input node for a message flow containing a publication node. Messages sent to this queue are processed by the message broker, then published to applications that have subscribed on the topic specified in the message.

Stream names are case sensitive.

After you type a new name, then save your changes, the name becomes available for selection in the drop-down list.

7. Select the WebSphere MQ message broker subscription point from which the service integration bus receives messages. If the subscription point you require is not on the list, click **other, please specify** then enter the name of the subscription point.

   The default subscription point is used if no value is specified.

   After you type a new name, then save your changes, the name becomes available for selection in the drop-down list.
8. Click **OK**.
9. Save your changes to the master configuration.
10. Optional: If you have enabled dynamic configuration updates, the changes take effect immediately. Otherwise, restart the application server.

Adding or modifying a WebSphere MQ link receiver channel:

How you can define the properties of the receiver channel on a WebSphere MQ link. This channel receives messages from the WebSphere MQ queue manager or queue-sharing group (known as the "gateway queue manager"). The receiver channel communicates with a WebSphere MQ sender channel on the gateway queue manager, and converts MQ format messages to service integration bus messages.

Before you begin

You have to know the name of the bus, and the messaging engine on the bus that contains the WebSphere MQ link with the receiver channel you intend to add or modify.

About this task

When you use the foreign bus connection wizard to connect a bus and a gateway queue manager to use point-to-point messaging or publish-subscribe messaging, the wizard does not ask you to set all possible properties of a WebSphere MQ link. If you did not choose to **Enable Service integration bus to WebSphere MQ message flow** in the foreign bus connection wizard, you have not yet defined a Websphere MQ receiver channel. If you did choose this option in the wizard, you have defined the receiver channel name, host name and port number, and the wizard has set the other properties to default values.

To add or modify a WebSphere MQ link receiver channel, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Receiver channel**.
2. In the content pane, either click **New** to add a new receiver channel, or click the name of a existing receiver channel that you want to modify.
3. Add or modify the properties of the channel.

   For information about all the properties that you can modify, see "WebSphere MQ link receiver channel [Settings]" on page 2208.

If you used the foreign bus connection wizard to create the channel, the wizard has used default values for the following properties:

Inbound nonpersistent message reliability
The acceptable reliability of message delivery for nonpersistent message flows from WebSphere MQ through this WebSphere MQ link, from Best effort to Reliable, in order of increasing reliability. By default, the value is “Reliable”.

Inbound persistent message reliability
The acceptable reliability of message delivery for inbound persistent message flows from WebSphere MQ through this WebSphere MQ link, from Reliable to Assured, in order of increasing reliability. By default, the value is “Assured”.

Prefer queue points local to this link's messaging engine
When this check box is selected, the link prefers to send inbound messages to available queue points of target queue destinations that are located on the same messaging engine as the link. By default the check box is selected, which corresponds to the behavior in WebSphere Application Server Version 6 and can make it easier to handle links in a mixed-version cell. If you clear the check box, preference is not given to local queue points and inbound messages are workload balanced across all available queue points of target queue destinations. This option (not to give preference to local queue points) is available only on links running on WebSphere Application Server Version 7.0 or later.

Initial state
Whether the receiver channel is started or stopped when the associated WebSphere MQ link is first started. Until started, the channel is unavailable. By default, the value is “Started”.

4. Click **OK**.
5. Save your changes to the master configuration.
6. Restart the application server.

Adding or modifying a WebSphere MQ link sender channel:

The WebSphere MQ link sender channel sends messages to the WebSphere MQ queue manager or queue-sharing group (known as the “gateway queue manager”). The sender channel communicates with a WebSphere MQ receiver channel on the gateway queue manager, and converts service integration bus messages to MQ format messages. Use this task to add a sender channel if you have not already defined one, or to fine tune the properties of an existing sender channel.

Before you begin

To be able to complete this task, you must know the name of the bus, and the messaging engine on the bus that contains the WebSphere MQ link with the sender channel that you intend to add or modify.

About this task

This task allows you to define a WebSphere MQ sender channel if you have not already done so when using the foreign bus connection wizard to create a foreign bus and link it to the WebSphere MQ network through a WebSphere MQ link. If you did not choose the **Enable WebSphere MQ to Service integration bus message flow** option when using the foreign bus connection wizard, you have not yet defined the sender channel.

If you did choose the **Enable WebSphere MQ to Service integration bus message flow** option in the wizard, you will already have defined the sender channel name, host name, communication port and transport chain. However, the wizard will have set the other properties of the sender channel to default values, in which case you might need to fine tune these properties using this task.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Sender channel**.
3. In the content pane, either click **New** to add a new sender channel, or click the name of a existing sender channel that you want to modify.
4. Add or modify the properties of the channel.

   For information about all the properties that you can modify, see “WebSphere MQ link sender channel [Settings]” on page 2214.

   If you used the foreign bus connection wizard to create the channel, the wizard has used default values for the following properties:

   Disconnect interval
   The time in seconds for which the sender channel waits for new messages to arrive on the transmission queue after sending a batch of messages. The channel disconnects after this interval, and must be restarted manually or by triggering. By default, the value is 900 seconds.

   Short retry count
   The maximum number of times that the sender channel tries to restart after a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then the long retry mechanism is invoked. By default, the value is 10.

   Short retry interval
   The number of seconds between attempts by the sender channel to restart after a communication or partner failure. By default, the value is 60 seconds.

   Long retry count
   The maximum number of times that the sender channel tries to restart after the short retry mechanism did not recover from a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then an error is logged and the channel is stopped. By default, the value is 999999999.

   Long retry interval
   The number of seconds between attempts by the sender channel to restart after the short retry mechanism did not recover from a communication or partner failure. By default, the value is 1200 seconds.

   Initial state
   Whether the sender channel is started or stopped when the associated WebSphere MQ link is first started. Until started, the channel is unavailable. By default, the value is “Started”.
5. Click **OK**.
6. Save your changes to the master configuration.
7. Restart the application server.

Modifying security for a WebSphere MQ link:

Securing access between a service integration bus and a WebSphere MQ Queue Manager.

About this task

When you create a new WebSphere MQ link, you can use the foreign bus connection wizard to enable security:

- If the WebSphere MQ queue manager requires a secure connection, you can set the WebSphere MQ receiver channel to accept only connections that have secure sockets layer (SSL) based encryption.
- If the local bus is secure, you can set the service integration bus inbound user ID to replace the user ID in messages from the WebSphere MQ queue manager, so that these messages are authorized to access their destinations.

  Inbound user ID

  If an inbound user ID is set, then all incoming messages will appear to have originated from that user ID. If the bus is security enabled then messages will appear authenticated as this user ID and have access to any resources that the user ID has access to.

  If an inbound user ID is not set, then messages will have the same user ID as in the WebSphere MQ message descriptor (MQMD) header of the WebSphere MQ message. These users will not be authenticated and therefore only have access to resources that require no authentication.

  Outbound user ID

  If an outbound user ID is set, then all outgoing messages will appear to have originated from that user ID (using the userid field of the MQMD)

  If an outbound user ID is not set, then messages will have the same user ID as in the original service integration bus message.

Use this task to secure the local and foreign bus that are part of a WebSphere MQ links configuration, and to secure an existing WebSphere MQ link that was not secured when it was first created.

For more general information about service integration bus security, see Securing service integration.

Procedure

1. Enable security on the service integration bus and the foreign bus representing the WebSphere MQ network. See Securing buses.
2. Secure the link between the buses - see Securing connections to a WebSphere MQ network.
3. Grant access to the local bus for users who will be sending messages to the foreign bus - see Securing buses.
4. Grant access to the foreign bus for users who will be sending messages to it - see Administering foreign bus roles.
5. Optional: Give users access to foreign or alias destinations that will forward messages to a foreign bus - see “Administering destination roles” on page 1991.

Viewing the status of a WebSphere MQ link and its sender and receiver channels:

You might want to view the status of a WebSphere MQ link or its components because you intend to stop the WebSphere MQ link and you want to see if there are any messages currently held on it, or you are about to delete the WebSphere MQ link and you have to verify there are no messages on it, or messages have not arrived at their expected destination and you want to check if they are being held, pending transmission over the WebSphere MQ link.

About this task

To view the status of a WebSphere MQ link, use the administrative console to complete the following steps.

Note: If you attempt to start a WebSphere MQ sender channel that is already in RUNNING state, the WebSphere Application Server administrative console might occasionally incorrectly report the channel status as INACTIVE. If this happens, you can ignore the error because the channel is still running. To return the channel status to RUNNING, stop and then restart the channel.

Procedure

1. In the navigation pane, click one of the following paths:
   - **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links**
   - **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links**
2. Click the WebSphere MQ link you want to view.
3. Click the **Runtime** tab. The content pane displays the status of the WebSphere MQ link.
4. Click the **Configuration** tab.
5. To view the status of the sender channel:
   a. Click **Sender channel**.
   b. Click the channel you want to view.
   c. Click the **Runtime** tab. The content pane displays status information for the sender channel. Some of the more important fields to check are:
      - The status of the channel.
      - Whether the message on the channel is in an indoubt state. If the message is in doubt, it has not been acknowledged by WebSphere MQ.
      - The number of messages in the current batch.
      - The number of batches that have been sent.
      - The time and date at which the channel was last started.
      - The time and date at which the last message was sent.
   d. Click **Saved batch status**. The content pane displays the saved status of message batches that have been saved for transmission to WebSphere MQ.
   e. If a batch is in an indoubt state, you can either commit or rollback the batch.
6. To view the status of the receiver channel:
   a. Return to the WebSphere MQ link page.
   b. Click **Receiver channel**.
   c. Click the **Runtime** tab. The content pane displays the status of the receiver channel.
   d. Click **Receiver channel connections**. The content pane displays the connections existing on the receiver channel, and their current status. You can stop connections if required, by selecting the check box next to the connection and clicking **Stop**.
   e. Return to the receiver channel page.
   f. Click **Saved batch status**. The content pane displays the saved status of message batches that have been received from WebSphere MQ.

What to do next

You can also view the status of subscriptions for a broker profile on the WebSphere MQ link: “Viewing the status of subscriptions for a WebSphere MQ link publish/subscribe broker profile.”

Viewing the status of subscriptions for a WebSphere MQ link publish/subscribe broker profile:

You can use the administrative console to view the status of subscriptions for a broker profile on a WebSphere MQ link.

Procedure

1. In the navigation pane, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles**

2. Click the broker profile containing the subscriptions you want to view.
3. Click the **Runtime** tab. The content pane displays the current number of subscriptions for the broker profile.
4. Click **Subscriptions**. The content pane displays the status of the subscriptions for the broker profile. Move the mouse pointer over a status icon to view hover help for that icon. You can remove the runtime subscriptions if required, by clicking **Unsubscribe**.

States of the WebSphere MQ link and its channels:

For each possible state of a WebSphere MQ link, this table lists the associated states of the link's sender and receiver channels.

State of WebSphere MQ link	WebSphere MQ link sender channel	WebSphere MQ link receiver channel
INACTIVE	Same as STOPPED for the WebSphere MQ link sender. If the administrator requests that the channel go into target state STOPPED it will transition into the STOPPED state. Similarly, an administrator starting the channel will cause it to transition into STANDBY state.	No network connection exists between the application server and the queue manager. If the attempts of a WebSphere MQ sender channel to establish a connection should succeed, it will become possible for messages to flow from the queue manager to the messaging engine. In this case the receiver channel will transition into RUNNING state.
STARTING	A transitional state. The channel should successfully pass through this into BINDING state with no intervention.	A transitional state. The channel should successfully pass through this into BINDING state with no intervention.
BINDING	A transitional state. The channel should successfully pass through this into RUNNING state with no intervention. The channel might transition into STOPPING state if a problem occurs.	A transitional state. The channel should successfully pass through this into RUNNING state with no intervention. The channel might transition into STOPPING state if a problem occurs.
INITIALIZING	A transitional state. The channel should successfully pass through this into STARTING state with no intervention.	A transitional state. The channel should successfully pass through this into STARTING state with no intervention.
RETRYING	A network connection to the queue manager has been lost. The channel attempts to reconnect. If the retry intervals are exhausted without successfully establishing the connection then the channel enters STOPPED state. If a connection is successfully reestablished, the channel enters INITIALIZING state.	Not applicable to receiver channel.

State of WebSphere MQ link	WebSphere MQ link sender channel	WebSphere MQ link receiver channel
STANDBY	When in this state, the sender channel is not network-connected to its WebSphere MQ counterpart receiver channel. It is waiting for a message to send before attempting to establish a connection. When a message arrives for transmission, the channel will transition into STARTING state and start the process of attempting to establish a network connection. The administrator may command the channel to transition into either INACTIVE or STOPPED state from this state.	Not applicable to receiver channel.
RUNNING	In this state a network connection has been established between application server and queue manager. Messages destined for the queue manager will be transmitted. Either attempting to stop the channel by using the administrative console, or the loss of the network connection will cause transition into the STOPPING state.	Network connection established between application server and queue manager. Messages destined for the messaging engine will be received. Attempting to stop the channel, or the loss of the network connection will cause a transition into STOPPING state.
STOPPING	A transitional state. The channel should transition into either RETRYING state or STOPPED state without intervention. If the channel has been placed in STOPPING state by the administrative request to become INACTIVE then it will transition into the STANDBY state. If the channel has been placed in this state by administrator request to stop, it will transition into the STOPPED state. If the channel has been placed in this state by a broken network connection it will transition into the RETRYING state, assuming that it has non-zero retry intervals or is otherwise STOPPED.	A transitional state. The channel will transition from this state into STOPPED without intervention.
STOPPED	No network connection exists between the application server and the queue manager. Messages destined for the queue manager will not be transmitted. To transition from this state requires the administrator to start the channel, this will place it in STANDBY state.	No network connection exists between the application server and the queue manager. Any attempt by a sender channel in the WebSphere MQ network to establish a connection will be rejected. Messages destined for the messaging engine will not be received. Administrator action is required to move the channel out of this state. Starting the channel will move it into INACTIVE state.

For information about the states of the channels in a WebSphere MQ network refer to the *Intercommunication* section of the WebSphere MQ information center, available from the WebSphere MQ library.

You can stop a WebSphere MQ link (and its sender and receiver channels) on a service integration bus, or you can stop individual sender or receiver channels. The following sections describe in more detail what happens when you transition between states.

Stopping a WebSphere MQ link

Stopping a WebSphere MQ link results in both the WebSphere MQ link sender and the WebSphere MQ link receiver channels being stopped:

- If a currently RUNNING WebSphere MQ link is stopped in state STOPPED with mode QUIESCE, the overall state of the WebSphere MQ link goes to STOPPED state. The WebSphere MQ link sender channel goes to STOPPED state. The WebSphere MQ link receiver channel goes to STOPPED state.
- If a currently RUNNING WebSphere MQ link is stopped in state INACTIVE with mode QUIESCE, the overall state of the WebSphere MQ link remains set to RUNNING. The WebSphere MQ link sender channel goes to STANDBY state. The WebSphere MQ link receiver channel goes to INACTIVE state. The WebSphere MQ sender channel will stop when convenient, as described below.
- If a currently RUNNING WebSphere MQ link is stopped in state STOPPED with mode FORCE, the overall state of the WebSphere MQ link goes to STOPPED state. The WebSphere MQ link sender channel goes to STOPPED state. The WebSphere MQ link receiver channel goes to STOPPED state.
- If a currently RUNNING WebSphere MQ link is stopped in state INACTIVE with mode FORCE, the overall state of the WebSphere MQ link remains set to RUNNING. The WebSphere MQ link sender goes to STANDBY state. The WebSphere MQ link receiver channel goes to INACTIVE state.

Stopping a WebSphere MQ link sender channel

Stopping a WebSphere MQ link sender stops only the WebSphere MQ link sender channel. (However, when the WebSphere MQ link sender channel is stopped, it communicates with the receiver channel in the WebSphere MQ network to say it is stopping, with the result that the receiver channel on WebSphere MQ stops and goes into inactive state.)

If a currently RUNNING WebSphere MQ link sender channel is stopped in state STOPPED, it goes to STOPPED state.

If a currently RUNNING WebSphere MQ link sender channel is stopped in state INACTIVE, it goes to STANDBY state.

Stopping a WebSphere MQ link receiver channel

Stopping a WebSphere MQ link receiver stops all receiver channel connections for that receiver.

If a currently RUNNING WebSphere MQ link receiver channel is stopped in state STOPPED, it goes to STOPPED state. The sender channel in the WebSphere MQ network will notice, either when it tries to next send some data, or when its heartbeat interval is reached and it tries to send a heartbeat flow, or when its disconnect interval expires and it attempts to close the session, that the WebSphere MQ link receiver in the service integration bus is in STOPPING state and itself stop and then enter a state of RETRYING. The WebSphere MQ link receiver will then go to STOPPED state, so preventing a sender channel in the WebSphere MQ network from establishing a session.

If a sender channel in the WebSphere MQ network is started while a WebSphere MQ link receiver channel in a service integration bus is in STOPPED state, the request fails with an error indicating that the WebSphere MQ link receiver channel is not available.

If a currently RUNNING WebSphere MQ link receiver channel is stopped in state INACTIVE, it goes to STOPPING state. The sender channel in the WebSphere MQ network will notice, either when it tries to next send some data, or when its heartbeat interval is reached and it tries to send a heartbeat flow, or when its disconnect interval expires and it attempts to close the session, that the WebSphere MQ link

receiver is in STOPPING state and itself stop and then enter a state of RETRYING. The WebSphere MQ link receiver will then go to INACTIVE state. The RETRYING sender channel in the WebSphere MQ network will then establish a session with the WebSphere MQ link receiver channel, at which point both channels will go to RUNNING state.

WebSphere MQ link receiver channel connection

Stopping a WebSphere MQ link receiver channel connection stops only that connection. Individual connections can only be stopped in target state INACTIVE.

Modes of a stopped WebSphere MQ link

When an active WebSphere MQ link receiver channel connection is stopped, the connection goes to a state of STOPPING. The sender channel in the WebSphere MQ network will notice, either when it tries to next send some data, or when its heartbeat interval is reached and it tries to send a heartbeat flow, or when its disconnect interval expires and it attempts to close the session, that the WebSphere MQ link receiver is in STOPPING state and itself stop and then enter a state of RETRYING.

The MODE has an effect on the stopping of channels.

1. QUIESCE, the channel stops when it is convenient for it to do so.

   For a WebSphere MQ link sender channel, the link sender goes to STANDBY or STOPPED state (depending on the stop state specified), this can occur either when it reaches the end of the current batch, or when it reaches a heartbeat interval.

   In the case of a WebSphere MQ link receiver, the link receiver goes to STOPPING state and then to INACTIVE or STOPPED state (depending on the stop state specified) when the sender in the WebSphere MQ network next attempts to communicate with it. Though, if a WebSphere MQ link receiver channel goes INACTIVE and the sender channel in the WebSphere MQ network goes to RETRYING, then as soon as a session is reestablished, both ends will go to RUNNING state.
2. FORCE, the channel stops immediately.

   For a WebSphere MQ link sender channel, the WebSphere MQ link sender goes to STANDBY or STOPPED state (depending on the stop state specified).

   For a WebSphere MQ link receiver, the WebSphere MQ link receiver goes to INACTIVE or STOPPED state (depending on the stop state specified). When the sender channel in the WebSphere MQ network next tries to communicate with the WebSphere MQ link receiver, it will either enter a state of RETRYING and reestablish a session with the WebSphere MQ link receiver, or go to a STOPPED state.

WebSphere MQ link sender channels can go to INDOUBT state (as can sender channels in a WebSphere MQ network). WebSphere MQ link receiver channels do not go to INDOUBT state.

While sending a batch of persistent messages, a WebSphere MQ link sender channel goes to a state of INDOUBT. When it commits the batch, it sends a commit request to the partner and waits for confirmation. When the partner sends a confirmation, the batch is finally committed and the sender channel is no longer in INDOUBT state. If the partner fails to send a confirmation flow, then the sender channel will remain in INDOUBT state.

INDOUBT batches can be COMMITTED or ROLLED back from the Saved Status panel for WebSphere MQ link sender channels.

Starting a WebSphere MQ link:

Change the state of a WebSphere MQ link from stopped to started.

Before you begin

You might want to tell the WebSphere MQ administrator that you are about to start this WebSphere MQ link.

About this task

To start a WebSphere MQ link, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click one of the following paths:
   - **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links**
   - **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links**
2. In the content pane, select the check box next to the WebSphere MQ link you want to start.
3. Click **Start**.

Results

If the WebSphere MQ link starts successfully, the status icon changes to indicate that the WebSphere MQ link is running.

Stopping a WebSphere MQ link:

You can change the state of a WebSphere MQ link from "Started" or "Running" to "Stopped". When you stop a WebSphere MQ link, all communication with the target WebSphere MQ network is stopped for both point-to-point and publication and subscription. Messages waiting for transmission are held on the service integration bus, and the MQ sender channel cannot start. If there is a publish/subscribe bridge on the WebSphere MQ link, its operations are stopped.

Before you begin

Inform the WebSphere MQ administrator that you are about to stop this WebSphere MQ link.

About this task

To stop a WebSphere MQ link, use the administrative console to complete the following steps. You can also stop either the sender or receiver channel on the WebSphere MQ link, while leaving the link itself running. See “Stopping the sender channel on a WebSphere MQ link” on page 570, and “Stopping the receiver channel on a WebSphere MQ link” on page 569.

Procedure

1. In the navigation pane, click one of the following paths:
   - **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links**
   - **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links**
2. In the content pane, select the check box next to the WebSphere MQ link that you want to stop.
3. In the **Stop mode** list, select "Quiesce" or "Force".
4. In the **Target state** list select "Inactive" or "Stopped".
5. Click **Stop**.

Results

The resultant state of the WebSphere MQ link, its sender and receiver channels, and the WebSphere MQ sender channel that is connected to the WebSphere MQ link receiver channel, depends on the options you chose:

- Stopping the WebSphere MQ link to target state INACTIVE causes the sender channel to go into state STANDBY and the receiver channel to go into state INACTIVE. The overall WebSphere MQ link status will be RUNNING.
- Stopping the WebSphere MQ link to target state STOP causes the sender channel to go into state STOPPED and the receiver channel to go into state STOPPED. The overall WebSphere MQ link status will be STOPPED.

These are the final states that the sender, receiver and WebSphere MQ link transition into. If you specify a mode of QUIESCE for either of the two final states, the channels and link might not transition into their final states immediately. Instead they temporarily transition through other states that are required to reach their final state.

For more details about stopped states of the WebSphere MQ link, see “States of the WebSphere MQ link and its channels” on page 564.

What to do next

You can restart the WebSphere MQ link by selecting the link again and clicking **Start**.

If the WebSphere MQ link is started and the sender channel is in the stop state, it goes into the standby state. The sender channel goes into negotiation with the partner receiver channel if it is started from the standby state. The sender channel then moves into the running state if the negotiation successful, or into the retry state if the negotiation fails. If messages are present when the WebSphere MQ link is started, the sender channel automatically goes into negotiation with its partner receiver channel and then moves to the running or retry state.

When the WebSphere MQ link is stopped, and not in doubt, you can delete the WebSphere MQ link and any associated publish/subscribe broker profiles and topic mappings as described in “Removing a foreign bus connection from a bus” on page 1917.

Stopping the receiver channel on a WebSphere MQ link:

You can stop the receiver channel on a WebSphere MQ link while leaving the link itself running.

Before you begin

Note: If you stop the receiver channel on a WebSphere MQ link, communication with the target WebSphere MQ network on that channel will cease for both point-to-point messaging and publishing and subscribing. Messages will be held at their transmission locations.

You might want to warn the administrator of the WebSphere MQ network that you are about to stop the channel.

About this task

If you stop a receiver channel, messages sent to the WebSphere MQ link engine are not received.

If a WebSphere MQ sender channel is started while an MQ link receiver channel is stopped, the request fails with an error indicating that the receiver channel is not available.

For more information about stopped states of the WebSphere MQ link and its channels, see “States of the WebSphere MQ link and its channels” on page 564.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Receiver channel**.
3. Select the check box next to the channel you want to stop.
4. In the **Stop mode** list, select Quiesce or Force.
5. In the **Target state** list, select Inactive or Stopped.
6. Click **Stop**.

Results

Stopping a receiver channel stops all the receiver channel connections for that receiver. The resultant state of the receiver channel, and the sender channel in the WebSphere MQ network with which it is communicating, depends on the options you choose:

Table 49. Stop modes. The table contains information about the target states and the corresponding stop modes for stopping the receiver channel on a WebSphere MQ link. There are two target states such as inactive and stopped, and there are two stop modes such as quiesce and force. The rows in the table represent the two target states, and the two stop modes are described in the two columns for each of the target state.

		Stop mode	
		Quiesce	Force
Target state	Inactive	The receiver channel moves to the stopping state and the data flow to the WebSphere MQ sender channel stops. When the WebSphere MQ sender channel next tries to communicate with the receiver channel the WebSphere MQ sender channel enters a state of retrying. The receiver channel then becomes inactive. The retrying WebSphere MQ sender channel then reestablishes a session with the receiver channel, and both channels become running.	The receiver channel immediately becomes inactive. When the WebSphere MQ sender channel next tries to communicate with the receiver channel, the WebSphere MQ sender channel enters a state of retrying. The retrying WebSphere MQ sender channel then reestablishes a session with the receiver channel, and both channels become running.
	Stopped	The receiver channel moves to the stopping state and the data flow to the WebSphere MQ sender channel stops. When the WebSphere MQ sender channel next tries to communicate with the receiver channel the WebSphere MQ sender channel enters a state of retrying. The receiver channel then becomes stopped, so preventing the WebSphere MQ sender channel from reestablishing a session. The WebSphere MQ sender channel itself then becomes stopped.	The receiver channel immediately becomes stopped. When the WebSphere MQ sender channel next tries to communicate with the receiver channel, the WebSphere MQ sender channel enters a state of retrying, and then becomes stopped itself.

Stopping the sender channel on a WebSphere MQ link:

You can stop the sender channel on a WebSphere MQ link while leaving the link itself running.

Before you begin

You might want to tell the WebSphere MQ network administrator that you are about to stop a channel.

About this task

When you stop the sender channel on a WebSphere MQ link, communication with the target WebSphere MQ network on that channel is stopped for both point-to-point messaging and publishing and subscribing. Messages are held at their transmission locations.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Sender channel**.
3. Select the check box next to the channel you want to stop.
4. In the **Stop mode** list, select Quiesce or Force.
5. In the **Target state** list, select Inactive or Stopped.
6. Click **Stop**.

Results

Only the sender channel is affected by this procedure. The resultant state of the sender channel depends on the options you chose:

Table 50. Stop modes. The table contains information about the target states and the corresponding stop modes for stopping the sender channel on a WebSphere MQ link. There are two target states such as inactive and stopped, and there are two stop modes such as quiesce and force. The rows in the table represent the two target states, and the two stop modes are described in the two columns for each of the target state.

		Stop mode	
		Quiesce	Force
Target state	Inactive	The sender channel becomes inactive either when it has finished processing its current batch, or when it reaches a heartbeat interval.	The sender channel immediately becomes inactive.
	Stopped	The sender channel becomes stopped either when it has finished processing its current batch, or when it reaches a heartbeat interval.	The sender channel immediately becomes stopped.

For more information about stopped states of the WebSphere MQ link and its channels, see “States of the WebSphere MQ link and its channels” on page 564.

Deleting a WebSphere MQ link publish/subscribe broker profile:

This describes how you delete a broker profile and all topic mappings on a WebSphere MQ link, which forms a publish/subscribe bridge between WebSphere Application Server and a WebSphere MQ network.

Before you begin

Before you start you have to know the names of the bus, messaging engine, and WebSphere MQ link that has the broker profile that you intend to delete. You should also consider informing the WebSphere MQ administrator that you are about to delete the connection to the message broker in the WebSphere MQ network.

If you also intend to delete the associated WebSphere MQ link, you need not complete this task. Refer instead to “Removing a foreign bus connection from a bus” on page 1917.

About this task

Deleting a broker profile is a three-stage operation to ensure both the application server and the WebSphere MQ network and its message brokers are synchronized after the deletion:

- Remove the subscriptions by unsubscribing the topic mappings on the broker profile.
- When the Runtime view is empty, delete the broker profile.
- If you have enabled dynamic configuration updates, the changes take effect immediately, otherwise restart the application server.

Note: If you remove the subscriptions but do not delete the broker profile, then the subscriptions are recreated when the server is restarted (because they are still present in the static configuration information for the WebSphere MQ link). These subscriptions are unrelated to the original subscriptions so this can lead to some messages in a publication flow being missing for subscribers on the target side of the bridge. For example, any messages published on an unsubscribed topic between the time the unsubscribe took place and the application server was restarted are not republished to the target side of the WebSphere MQ link.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name***.
3. Remove the subscriptions:
   a. Click the **Runtime** tab.
   b. Click **Subscriptions**.
   c. Click **Unsubscribe** to remove all the subscriptions listed. When an unsubscribe command is sent to the message broker in the WebSphere MQ network, the relevant topic mapping is put into an indoubt state until the unsubscribe is confirmed when the topic mapping is deleted. After the unsubscribe is confirmed the topic mapping is no longer shown in the runtime view. You might have to refresh the runtime view for all subscriptions to be shown as removed.
4. Delete the broker profile:
   a. Return to the **Publish/subscribe broker profiles** page.
   b. Select the check box next to the broker profile you want to delete.
   c. Click **Delete**.
5. Save your changes to the master configuration.
6. If you have enabled dynamic configuration updates, the changes take effect immediately. Otherwise, restart the application server.

Deleting a topic mapping on a WebSphere MQ link publish/subscribe broker profile:

This describes how you delete a topic mapping on a broker profile on a WebSphere MQ link, which forms part of a publish/subscribe bridge between WebSphere Application Server and WebSphere MQ.

Before you begin

Before you start you have to know the bus name, messaging engine name, WebSphere MQ link name and broker profile name on which you intend to delete the topic mapping.

Note:

- If you intend to delete *all* topic mappings before deleting the broker profile or the WebSphere MQ link, you can avoid restarting the application server more than once by following the steps given in “Deleting a WebSphere MQ link publish/subscribe broker profile” on page 571.

- Deleting a topic mapping is a two-stage operation to ensure both the WebSphere Application Server (base) and the message brokers in the WebSphere MQ network are synchronized once the deletions have taken place:
  1. Delete the topic mapping (see below).
  2. When you have deleted the topic mappings, the changes take effect immediately if you have enabled dynamic configuration updates, otherwise restart the application server.

About this task

To delete a topic mapping, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name* -> [Additional Properties] Topic mappings**
2. Select the check box next to the topic mapping you want to delete.
3. Click **Delete**.
4. Save your changes to the master configuration.

What to do next

Any topic mappings that you have deleted are automatically cleaned up by the publish/subscribe bridge when the application server is restarted.

Using a WebSphere MQ server to integrate WebSphere MQ queues into a bus

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. For interoperation with WebSphere Application Server Version 7.0 or later, the version of WebSphere MQ must be WebSphere MQ for z/OS Version 6 or later, or WebSphere MQ (distributed platforms) Version 7 or later.

Before you begin

WebSphere Application Server can interoperate with WebSphere MQ in any of the following ways:

- Using the WebSphere MQ messaging provider
- Using a WebSphere MQ link
- Using a WebSphere MQ server

Each type of interoperation is designed for different situations, and provides different advantages. For information about the differences between these approaches, see “Interoperating with a WebSphere MQ network” on page 545.

Decide which method to use to configure these resources. You can configure WebSphere MQ server resources by using the administrative console as described in this task, or by using the “SIBAdminCommands: WebSphere MQ server administrative commands for the AdminTask object” on page 2352.

About this task

To set up and use a WebSphere MQ server, you configure the server properties, add the server to a service integration bus as a bus member, and create a WebSphere MQ queue-type destination. Destinations that are assigned to a WebSphere MQ server bus member can also be mediated.

You add the WebSphere MQ server as a bus member so that messaging engines on the bus can access queues on the target WebSphere MQ system. If your WebSphere MQ server is connected to a queue-sharing group, your bus applications can access shared queues on the target installation.

Notes:

- You can configure a WebSphere MQ server to connect to a WebSphere MQ queue manager by using either a bindings mode or a client mode connection. To use client mode with WebSphere MQ for z/OS, you need an additional product called the Client Attach Facility.
- You should configure the queues on the WebSphere MQ network as "shareable". This allows multiple server instances to get messages from the queues.

Procedure

1. Create a WebSphere MQ server definition and configure the server properties.
2. Add the new WebSphere MQ server as a member of a bus so that messaging engines on the bus can access queues on the target WebSphere MQ installation.
3. Create a WebSphere MQ queue type destination for the new bus member and assign it to a WebSphere MQ queue.
4. Optional: Mediate the new destination by using the WebSphere MQ queue as the mediation point.

Creating a WebSphere MQ server definition:

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. To create a WebSphere MQ server definition, you use the administrative console to define the server connection and quality of service properties.

Before you begin

Decide which method to use to configure these resources. You can create a new WebSphere MQ server definition by using the administrative console as described in this task, or by using the "createSIBWMQServer command" on page 2352.

About this task

A WebSphere MQ server definition defines the connection to an underlying WebSphere MQ queue manager or queue-sharing group and the associated queues.

When you subsequently add the server as a member of a service integration bus, you can optionally override the server connection settings with the bus connection settings. This means that you can create a WebSphere MQ server definition that is specific to a bus, yet reusable in a multiple bus topology.

Procedure

1. Start the administrative console.
2. Complete either of the following sub-steps:
   a. Navigate to **Servers -> New server**, choose a server type of "WebSphere MQ server", then click **Next**.
   b. Navigate to **Servers -> Server Types -> WebSphere MQ servers**, then click **New**.

   The "WebSphere MQ server [Settings]" on page 2254 form is displayed.
3. Complete the fields as required.

   For more information, refer to the "WebSphere MQ server [Settings]" on page 2254 form and the following notes:

   Name
   The name that you use for this WebSphere MQ server definition must be unique.

UUID
This identifier is assigned automatically when you create a new WebSphere MQ server definition.

Use bindings transport mode if available
To connect to a WebSphere MQ queue manager or queue-sharing group in bindings mode, WebSphere Application Server needs to know where to load native libraries from. This information is stored in the **Native library path** property of the WebSphere MQ messaging provider. If you want to use a direct binding to WebSphere MQ, rather than a TCP/IP network connection, select this option and configure the **Native library path** property as described in "Configuring the WebSphere MQ messaging provider with native libraries information" on page 723.

If you are using Resource Access Control Facility (RACF®) as the security manager on your WebSphere MQ for z/OS system, and using bindings transport mode, you must specify in uppercase characters the user names and passwords for authentication aliases. If you are using RACF and client transport mode, you can specify the user names and passwords in either upper or lowercase characters.

Test connection
After you have configured the Connection properties, click this button to test the connection to WebSphere MQ.

Trust user identifiers received in messages
Select this option if you do not want the user IDs in messages to be overwritten with the administrative name of the WebSphere MQ server.

JAAS - J2C authentication data
This item is not available until after the WebSphere MQ server definition has been created.

4. Click **OK** to confirm.
5. Save your changes to the master configuration.
6. Restart the application server.

What to do next

You are now ready to add the new WebSphere MQ server as a member of a bus.

Modifying a WebSphere MQ server definition:

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This topic describes how to modify a WebSphere MQ server connection and quality of service properties.

Before you begin

Decide which method to use to configure these resources. You can modify a WebSphere MQ server by using the administrative console as described in this task, or by using the "modifySIBWMQServer command" on page 2355.

About this task

A WebSphere MQ server definition defines the connection to an underlying WebSphere MQ queue manager or queue-sharing group and the associated queues.

When you modify a WebSphere MQ server definition, you do not change any configuration values previously inherited from this WebSphere MQ server by existing bus members. For example, suppose you create a WebSphere MQ server with the port number 1234, then add the server to a bus and specify that server port number. If you subsequently modify the WebSphere MQ server port number to 2345, the bus member you previously created is not affected and still has the port number 1234.

Procedure

1. Start the administrative console.
2. Navigate to **Servers -> Server Types -> WebSphere MQ servers -> *server_name***. The “WebSphere MQ server [Settings]” on page 2254 form is displayed.
3. Make modifications as required.

   For more information, refer to the “WebSphere MQ server [Settings]” on page 2254 form and the following notes:

   UUID
   This identifier is assigned automatically when you create a new WebSphere MQ server definition.

   Use bindings transport mode if available
   To connect to a WebSphere MQ queue manager or queue-sharing group in bindings mode, WebSphere Application Server needs to know where to load native libraries from. This information is stored in the **Native library path** property of the WebSphere MQ messaging provider. If you want to use a direct binding to WebSphere MQ, rather than a TCP/IP network connection, select this option and configure the **Native library path** property as described in “Configuring the WebSphere MQ messaging provider with native libraries information” on page 723.

   If you are using Resource Access Control Facility (RACF) as the security manager on your WebSphere MQ for z/OS system, and using bindings transport mode, you must specify in uppercase characters the user names and passwords for authentication aliases. If you are using RACF and client transport mode, you can specify the user names and passwords in either upper or lowercase characters.

   Test connection
   After you have configured the Connection properties, click this button to test the connection to WebSphere MQ.

   Trust user identifiers received in messages
   Select this option if you do not want the user IDs in messages to be overwritten with the administrative name of the WebSphere MQ server.
4. Click **OK** to confirm.
5. Save your changes to the master configuration.
6. Restart the application server.

Deleting a WebSphere MQ server definition:

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This topic describes how to delete a WebSphere MQ server definition.

Before you begin

Decide which method to use to configure these resources. You can delete a WebSphere MQ server by using the administrative console as described in this task, or by using the “deleteSIBWMQServer command” on page 2360.

Ensure that no application is putting messages to the bus members located on the WebSphere MQ server.

Inform the WebSphere MQ administrator that the WebSphere MQ server is about to be deleted and therefore will no longer interoperate with its WebSphere MQ queue manager or queue-sharing group in the WebSphere MQ network.

About this task

When you delete a WebSphere MQ server definition, the deletion process also modifies the following associated resources:

- It deletes all WebSphere MQ server bus members that were created when the server was added to service integration buses.
- It unmediates all destinations that were assigned to those bus members.
- It removes all queue points that were assigned to those bus members.

Deleting a WebSphere MQ server definition does not affect the associated queue managers, queue-sharing groups, queues or messages on your WebSphere MQ network.

Procedure

1. Start the administrative console.
2. Navigate to **Servers -> Server Types -> WebSphere MQ servers**. The "WebSphere MQ servers [Collection]" on page 2253 form is displayed.
3. Select the check box next to the WebSphere MQ server you want to delete.
4. Click **Delete**. If the processing completes successfully, the list of WebSphere MQ servers is updated. Otherwise, an error message is displayed.
5. Save your changes to the master configuration.
6. Restart the application server.

Adding a WebSphere MQ server as a member of a bus:

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. A WebSphere MQ server bus member is used as a bus member for assigning queue points and mediation points to WebSphere MQ queues.

Before you begin

Get details of the client connection from your WebSphere MQ administrator.

Ensure that the WebSphere MQ server has been configured, that the bus has been defined and that the server is not already a member of the bus.

Decide which method to use to configure these resources. You can add the WebSphere MQ server as a bus member by using the administrative console as described in this task, or by using the "addSIBusMember command" on page 2268.

About this task

When you add a WebSphere MQ server to one or more buses, messaging engines on these buses can access queues on the target WebSphere MQ installation. When you make the server a bus member, you can override the server connection settings with settings that are specific to the new bus member. This can be useful in a multiple bus topology.

Procedure

1. Start the administrative console.
2. Navigate to the list of bus members for the bus to which you are adding the WebSphere MQ server. Click **Service integration -> Buses -> *bus_name* -> [Topology] Bus members**.
3. Click **Add**. The "Add a new bus member" wizard is displayed.
4. Select the WebSphere MQ server to add to the bus:
   a. Select **WebSphere MQ server**.
   b. From the drop-down list, select the server to add.
   c. Click **Next**.
5. Specify the virtual queue manager name.

When sending messages to WebSphere MQ, the WebSphere MQ gateway queue manager sees the bus as a remote queue manager. The virtual queue manager name is the name that is passed to WebSphere MQ as the name of this remote queue manager. The default value is the name of the bus. If this value is not a valid name for a WebSphere MQ queue manager, or if another WebSphere MQ queue manager already exists that has the same name, then replace the default value with another value that is a valid and unique name for a WebSphere MQ queue manager. To be valid, the name must meet the following criteria:

- It must contain between 1 and 48 characters.
- It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

6. Optional: To override the server connection settings, select the **Override WebSphere MQ server connection properties** check box.

   When you select this option, the connection properties for the server are made available so that you can change them to settings that are specific to this bus member. For more information about these connection properties, see “WebSphere MQ server bus member [Settings]” on page 2250.
7. Optional: If you have changed the server connection settings, you can click **Test connection** to test the connection to the associated WebSphere MQ network.
8. Click **Next**.
9. Click **Finish** to confirm.
10. Save your changes to the master configuration.

What to do next

You are now ready to create a WebSphere MQ queue-type destination for the new bus member.

Modifying a WebSphere MQ server bus member definition:

A WebSphere MQ server bus member is used for assigning queue points and mediation points to WebSphere MQ queues. This topic describes how to modify the attributes of a WebSphere MQ server bus member.

Before you begin

Decide which method to use to configure these resources. You can modify a WebSphere MQ server bus member by using the administrative console as described in this task, or by using the “modifySIBWMQServerBusMember command” on page 2361.

About this task

A WebSphere MQ server provides a direct client connection between a service integration bus and queues on a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command modifies the attributes of a WebSphere MQ server bus member.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> Buses -> *bus_name* -> [Topology] Bus members -> *member_name***. The “WebSphere MQ server bus member [Settings]” on page 2250 form is displayed.
3. Make modifications as required. For more information, refer to the “WebSphere MQ server bus member [Settings]” on page 2250 form.
4. After you have configured the **Connection settings**, click **Test connection** to test the connection toWebSphere MQ.
5. When you have made all the changes that you require, click **OK** to confirm.
6. Save your changes to the master configuration.

7. Restart the application server.

Deleting a WebSphere MQ server bus member definition:

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. You cannot delete a WebSphere MQ server bus definition directly. Instead, when you delete a specified WebSphere MQ server, you remove any associated WebSphere MQ server bus members, as well as assigned queue points and mediation points.

Before you begin

Decide which method to use to configure these resources. You can delete a WebSphere MQ server by using the administrative console as described in “Deleting a WebSphere MQ server definition” on page 576, or by using the “deleteSIBWMQServer command” on page 2360.

About this task

When you delete a WebSphere MQ server definition, the deletion process also modifies the following associated resources:

- It deletes all WebSphere MQ server bus members that were created when the server was added to service integration buses.
- It un-mediates all destinations that were assigned to those bus members.
- It removes all queue points that were assigned to those bus members.

Deleting a WebSphere MQ server definition does not affect the associated queue managers, queue-sharing groups, queues or messages on your WebSphere MQ network.

Creating a queue-type destination and assigning it to a WebSphere MQ queue:

You can use the administrative console to create a queue-type destination and assign it to a WebSphere MQ queue. Select the WebSphere MQ server to host the queue, then specify the WebSphere MQ queue to be hosted.

Before you begin

Get the name of the WebSphere MQ queue from your administrator, and ensure that the following configuration is established:

- The WebSphere MQ server is added as a member of a bus
- The WebSphere MQ queue for the queue point exists
- The WebSphere MQ administrator has set the queue attributes to “shareable”

Note: A shareable queue can be accessed by more than one service integration application.

Decide which method to use to configure these resources. You can create a bus destination by using the administrative console as described in this task, or by using the “createSIBDestinations command” on page 2325.

About this task

After you have added a WebSphere MQ server as a bus member, you can create a queue-type destination on the bus member that uses a WebSphere MQ queue as a queue point. This configuration enables service integration applications to send messages to and receive messages from that queue.

Procedure

1. Start the administrative console.
2. Navigate to the list of destinations for the appropriate bus.

   Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations**.

   The "Destinations [Collection]" on page 2053 form is displayed.
3. Click **New**. The "Create a new destination" panel is displayed.
4. Select **Queue** as the destination type, then click **Next**. The "create a new queue" wizard is displayed.
5. Set the queue attributes. Enter the name that you want WebSphere Application Server to use to refer to the associated WebSphere MQ queue, and (optionally) a description of the queue.
6. Assign the queue to the bus member that is to store and process the messages for the queue.

   Select a WebSphere MQ server bus member from the list of available bus members.
7. Set the WebSphere MQ queue point attributes:
   a. Specify a value in the **WebSphere MQ queue name filter** field, then click **Go**.

      The wizard automatically discovers available WebSphere MQ queues. However, some WebSphere MQ topologies have many thousands of queues defined to a queue manager. Use this filter to limit the number of queues that are listed.

      The default filter value is an asterisk (*). If this value (or no value) is set then all queues, or all queues of a specific type (based on any queue type custom property that is set), are listed. Any other value that you specify must meet the following criteria:

      - It must contain between 1 and 48 characters.
      - It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

      You can also use the wildcard character (*) with other text. For example, if you enter a value of PAYROLL*, then all available queues with names that start with PAYROLL are displayed.

   b. Specify a WebSphere MQ queue name.

      Select a queue name from the filtered list. If the list does not include the queue that you want, select the last entry in the list labeled **other, please specify**. A text entry box is displayed next to the drop-down list. Type the queue name into the text entry box.

      If the queue is found on the remote WebSphere MQ system, the properties of the queue as defined within WebSphere MQ are displayed as read-only fields. This should help you to confirm that you have found the queue that you want, and that it is configured as you intend. If the queue is not found, these read-only fields are removed from view.

   c. Specify the reliability levels that you require when inbound nonpersistent and inbound persistent WebSphere MQ messages are converted to service integration format messages.

      Applications receive messages direct from the specified WebSphere MQ queue, so in general the reliability level for a message is of no interest to the receiver because the message has already been delivered successfully. However, the message is converted to a service integration format message (and typically to a JMS format service integration message) as it is received, and this option specifies the reliability level for the service integration format message. For information about the available reliability levels, see "WebSphere MQ queue points [Settings]" on page 2241.

   d. Specify whether you want WebSphere MQ to include an MQRFH2 message header when sending messages to the queue.

      The MQRFH2 header stores service integration messaging information that does not have a corresponding WebSphere MQ message header field. When a message is sent to the destination, service integration instructs WebSphere MQ to write the message to the queue. This option specifies whether service integration instructs WebSphere MQ to write the message with an MQRFH2 header.

      If the consumer of the message is a JMS application running in WebSphere MQ or service integration, or a WebSphere MQ XMS application, or a WebSphere MQ MQI application that

expects an MQRFH2 header, select this option. If the consumer is a WebSphere MQ MQI application that does not expect an MQRFH2 header, do not select this option.

8. Click **Next**.
9. Click **Finish** to confirm queue creation.

Results

You have created a queue-type destination with a WebSphere MQ queue point.

What to do next

You are now ready to (optionally) mediate the new destination by using the WebSphere MQ queue as the mediation point.

Mediating a destination by using a WebSphere MQ queue as the mediation point:

Mediate a destination by using the administrative console to specify a WebSphere MQ server bus member where the mediation point is to be assigned, and a WebSphere MQ queue to use as the mediation point where messages are stored. To mediate the destination using a service integration mediation, you must also specify a second bus member (not a WebSphere MQ server) to use as the mediation execution point and process the messages.

Before you begin

Decide which method to use to configure these resources. You can mediate a destination by using the administrative console as described in this task, or by using the “mediateSIBDestination command” on page 2335.

Before performing this task, ensure that the following resources exist:

- The mediation that you want to apply to the destination.
- The WebSphere MQ server bus member where the mediation point is to be assigned.
- The WebSphere MQ queue to use as the mediation point, with the queue attributes set to `shareable`.
- For a service integration mediation, a second bus member (not a WebSphere MQ server bus member) to use as the mediation execution point where the mediation code runs.

Note: The queue manager on the WebSphere MQ network does not have to be available when you complete this task, but the destination is not usable until the queue manager becomes available.

About this task

You can mediate a destination with a WebSphere MQ mediation point. This ensures that messages arriving at the designated WebSphere MQ queue are mediated. In this scenario, the mediated messages are delivered to the queue point, or to another destination that is determined by the default forward routing path destination, or by the mediation code. The mediation can be hosted by service integration, or hosted by WebSphere MQ.

Procedure

1. Start the administrative console.
2. Navigate to the list of destinations for the appropriate bus. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations**.
3. Select the check box beside the destination to mediate, then click **Mediate**. The Mediation wizard is displayed.
4. Step 1: Select mediation.

To mediate the destination by using a mediation hosted by service integration:

a. Select **The mediation to apply to this destination**.

b. From the drop-down list, select the mediation.

c. Click **Next**.

To mediate the destination by using a WebSphere MQ program (for example, a WebSphere MQ flow):

a. Select **Externally mediated**.

b. Click **Next**.

5. Step 2: Assign the mediation to a bus member.

   When a mediation is assigned to a WebSphere MQ server bus member, you need a separate bus member that is not a WebSphere MQ server to act as the mediation execution point and process the messages.

   a. From the drop-down list, select the WebSphere MQ server bus member where the mediation point is to be assigned.

   b. Optional: If you are using a service integration mediation, select the bus member where the mediation is to run.

      For a mediation hosted by service integration, select a bus member from the list box that is labeled **Select a bus member where the mediation will run**. If you are using an external mediation, by definition it does not run in a bus member.

   c. Click **Next**.

6. Optional: If the mediation point is a WebSphere MQ queue, set the WebSphere MQ mediation point attributes.

   Note: This step is only displayed if you assigned the mediation point to a WebSphere MQ queue in the previous step.

   a. Specify a value in the **WebSphere MQ queue name filter** field, then click **Go**.

      The wizard automatically discovers available WebSphere MQ queues. However, some WebSphere MQ topologies have many thousands of queues defined to a queue manager. Use this filter to limit the number of queues that are listed.

      The default filter value is an asterisk (*). If this value (or no value) is set then all queues, or all queues of a specific type (based on any queue type custom property that is set), are listed. Any other value that you specify must meet the following criteria:

      - It must contain between 1 and 48 characters.
      - It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

      You can also use the wildcard character (*) with other text. For example, if you enter a value of PAYROLL*, then all available queues with names that start with PAYROLL are displayed.

   b. Specify a WebSphere MQ queue name.

      Select a queue name from the filtered list. If the list does not include the queue that you want, select the last entry in the list labeled **other, please specify**. A text entry box is displayed next to the drop-down list. Type the queue name into the text entry box.

      If the queue is found on the remote WebSphere MQ system, the properties of the queue as defined within WebSphere MQ are displayed as read-only fields. This should help you to confirm that you have found the queue that you want, and that it is configured as you intend. If the queue is not found, these read-only fields are removed from view.

   c. Specify the reliability levels that you require when inbound nonpersistent and inbound persistent WebSphere MQ messages are converted to service integration format messages.

      Mediations receive messages direct from the specified WebSphere MQ queue, so in general the reliability level for a message is of no interest to the mediation because the message has already been delivered successfully. However, the message is converted to a service integration format message (and typically to a JMS format service integration message) as it is received, and this

option specifies the reliability level for the service integration format message. For information about the available reliability levels, see “WebSphere MQ queue points [Settings]” on page 2241.

d. Specify whether you want WebSphere MQ to include an MQRFH2 message header when sending messages to the queue.

   The MQRFH2 header stores service integration messaging information that does not have a corresponding WebSphere MQ message header field. When a message is sent to the destination, service integration instructs WebSphere MQ to write the message to the queue. This option specifies whether service integration instructs WebSphere MQ to write the message with an MQRFH2 header.

   If the consumer of the message (in this case, the mediation) is a JMS application running in WebSphere MQ or service integration, or a WebSphere MQ XMS application, or a WebSphere MQ MQI application that expects an MQRFH2 header, select this option. If the mediation is a WebSphere MQ MQI application that does not expect an MQRFH2 header, do not select this option.

e. Click **Next**.

7. Check the summary of your selections, then click **Finish** to confirm mediation of the destination.

Results

You have mediated a destination by using a WebSphere MQ queue as the mediation point.

Enabling WebSphere Application Server Version 5.1 JMS usage of messaging resources in later versions of the product

To enable JMS applications developed for WebSphere Application Server Version 5.1 to use messaging resources of the default messaging provider, a WebSphere MQ client link is created on the node of the later version of the product. Each WebSphere MQ client link presents itself as a queue manager and transforms between the WebSphere MQ client protocols used by Version 5.1 and the protocols used by the default messaging provider in later versions.

Before you begin

Throughout this topic, the abbreviation “Version 5.1” refers to “WebSphere Application Server Version 5.1”. For example, “Version 5.1 JMS resources” refers to JMS resources provided by WebSphere Application Server Version 5.1.

This task refers to the *default messaging provider*. For related information, see “Managing messaging with the default messaging provider” on page 515.

JMS connectivity between the Version 5.1 messaging provider and the default messaging provider in later versions of the product is enabled and managed by a *WebSphere MQ client link*. This does not mean that a WebSphere MQ system is involved. The Version 5.1 messaging provider uses WebSphere MQ client protocols, and is therefore handled as if it were a WebSphere MQ client by the default messaging provider in later versions of the product. The WebSphere MQ client link is provided only for use with JMS applications developed for WebSphere Application Server Version 5.1. Moreover, this JMS connectivity is only intended as an aid to migration from the Version 5.1 messaging provider to the default messaging provider of later versions. For more information about migrating from the Version 5.1 messaging provider, see Migrating from WebSphere Application Server Version 5 embedded messaging.

Applications running in later versions can use the messaging resources of the Version 5.1 messaging provider without any need for a WebSphere MQ client link.

About this task

The following figure shows a JMS application running on Version 5.1 and using JMS resources provided by the default messaging provider on a Version 7.0 node. The JMS queue hosted by Version 5.1 is backed by a service integration bus queue, which is normal for a JMS queue hosted by Version 7.0, but there is no configured link between the Version 5.1 JMS queue and the bus queue. The JMS application communicates with the bus queue through the WebSphere MQ client link and the messaging engine. To send messages to the bus queue or receive messages from the queue, the JMS application opens a connection on the WebSphere MQ client link. This is all invisible to the JMS application, but can be displayed and managed by the administrator.

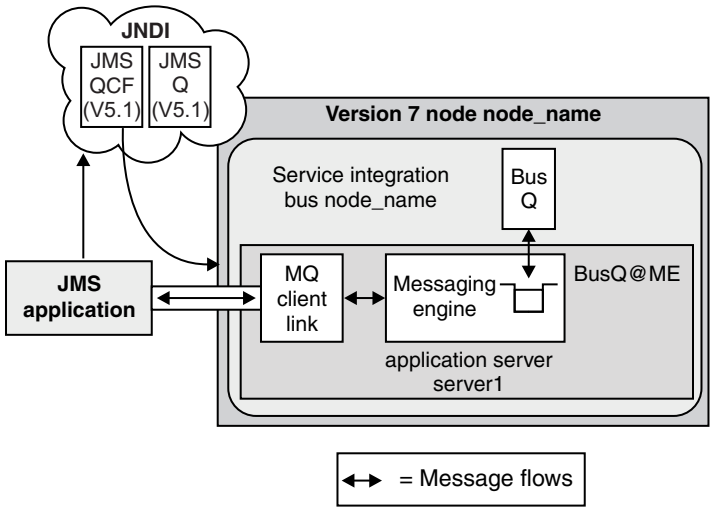


Figure 3. WebSphere Application Server Version 5.1 JMS application scenario

Procedure

- Configure a node on the later version to support Version 5.1 applications that use JMS resources. If you want a node on the later version to provide JMS destinations for use by applications running on Version 5.1, complete the following steps:
  1. Create an application server. You can use an existing application server on the node; for example, an application server onto which a Version 5.1 application is to be deployed.
  2. Create a service integration bus. You can use an existing bus.
  3. Add the application server as a bus member. This automatically creates a messaging engine on the application server.
  4. Create a WebSphere MQ client link on the messaging engine. Specify the following property values:

     Name This can be any name that is useful for your administrative purposes. It is not used by the application environment.

     MQ channel name
     This is the name of the channel for the WebSphere MQ client link, used to flow messages between the application that is running on Version 5.1 and the bus. This name must match the receiving channel name configured for Version 5.1:

     `WAS.JMS.SVRCONN`

     This is the default value shown when you first display the WebSphere MQ client link settings panel.

Queue manager name
This is the virtual queue manager name that is associated with the messaging engine, and by which the messaging engine is known to applications running on Version 5.1 . Type the queue manager name in the form:

`WAS_node_name_server_name`

Where:

node_name
is the name of the node on the later version of the product.

server_name
is the name of the application server.

The correct value is shown by default when you first display the WebSphere MQ client link settings panel.

Default queue manager
Select this check box if you want the WebSphere MQ client link to be used as the default for applications that cannot find a suitable WebSphere MQ client link to use.

If an application running on Version 5.1 specifies that it is to connect to a non-default queue manager name, you can configure a WebSphere MQ client link with that queue manager name. If a WebSphere MQ client link cannot be found with the required queue manager name, the connection is rejected. Alternatively, you can select this option on another WebSphere MQ client link, which is used instead of rejecting the connection.

5. Define a port called JMSSERVER_QUEUED_ADDRESS on the application server. The port number must be the same used by the SIB_MQ_ENDPOINT_ADDRESS port.

   Specify the following property values:

   Port name
   For **Well-known Port**, select JMSSERVER_QUEUED_ADDRESS

   Host Type the IP address, domain name server (DNS) host name with domain name suffix, or the short DNS host name of the node of the later version of the product.

   Port Type the port number used by the SIB_MQ_ENDPOINT_ADDRESS port. By default, this is 5558.

- Configure a managed node on the later version of the product to support applications running on Version 5.1 that use JMS resources. If you want a managed node to provide JMS destinations for use by applications running on Version 5.1, complete the following steps:
  1. Create an application server. Specify the name `jmsserver`.
  2. Create a service integration bus. You can use an existing bus.
  3. Add the application server as a bus member. This automatically creates a messaging engine on the application server.
  4. Create a WebSphere MQ client link on the messaging engine. Specify the following property values:

     Name This can be any name that is useful for your administrative purposes. It is not used by the application environment.

     MQ channel name
     This is the name of the channel for the WebSphere MQ client link, used to flow messages between the application running on Version 5.1 and the bus. This name must match the receiving channel name configured for Version 5.1:

     `WAS.JMS.SVRCONN`

     This is the default value shown when you first display the WebSphere MQ client link settings panel.

Queue manager name
This is the virtual queue manager name that is associated with the messaging engine, and by which the messaging engine is known to applications running on Version 5.1. Type the queue manager name in the form:

`WAS_node_name_jmsserver`

Where:

node_name
is the name of the node on the later version of the product.

The correct value is shown by default when you first display the WebSphere MQ client link settings panel.

Default queue manager
Select this check box if you want the WebSphere MQ client link to be used as the default for applications that cannot find a suitable WebSphere MQ client link to use.

If an application developed for WebSphere Application Server Version 5.1 specifies that it is to connect to a non-default queue manager name, you can configure another WebSphere MQ client link with that queue manager name. If a WebSphere MQ client link cannot be found with the required queue manager name, the connection is rejected. Alternatively, you can select this option on a WebSphere MQ client link, which is used instead of rejecting the connection.

5. Define a port called JMSSERVER_QUEUED_ADDRESS on the application server. The port number must be the same used by the SIB_MQ_ENDPOINT_ADDRESS port.

   Specify the following property values:

   Port name
   For **Well-known Port**, select JMSSERVER_QUEUED_ADDRESS

   Host Type the IP address, domain name server (DNS) host name with domain name suffix, or the short DNS host name of the node on the later version of the product.

   Port Type the port number used by the SIB_MQ_ENDPOINT_ADDRESS port. By default, this is 5558.

- If the application looks up JMS resources in JNDI on the application server, configure the JMS resources on the application server of the later version as Version 5.1 default messaging JMS resources.
  1. For each JMS queue destination that the application uses, create a V5 default messaging provider queue destination.
  2. For each JMS queue destination that the application uses, create a bus destination with the same name. Assign the bus destination to a bus member in the same bus as the `jmsserver` bus member.You must also create an alias destination with an identifier `WQ_<destination_name>`, that points to the service integration destination that has been created. The WQ_ prefix is needed because all destination names are prefixed with WQ_. If you are manually migrating the WebSphere JMS provider resources, you also have to create the “WQ_” queues.
  3. Configure JMS connection factories as Version 5.1 default messaging queue connection factories and topic connection factories.
- If the application looks up JMS resources outside the JNDI on the application server, configure the JMS connection factory to point to the node on the later version of the product.

Results

The application running on Version 5.1 can continue to access the JMS resources on the later level of the product, which are now implemented through the default messaging provider, as shown in the figure Figure 3 on page 584. The JMS application communicates with the Version 5.1 JMS resources through the WebSphere MQ client link and the messaging engine. This is invisible to the JMS application. The JMS

resources, a JMS queue connection factory, shown as JMS QCF(V5), and a JMS queue, shown as JMS Q(V5), are managed as Version 5.1 default messaging JMS resources. The new bus queue, shown as JMS Q, is managed as a resource of the service integration bus. Messages for JMS Q are stored and processed by the message point for the associated bus destination, a queue shown as Bus Q. The WebSphere MQ client link presents itself as a queue manager and transforms between the WebSphere MQ client protocols used by JMS applications developed for WebSphere Application Server Version 5.1 and the protocols used by messaging engines on the later version.

Creating a WebSphere MQ client link

Use this task to create a link that enables a messaging engine to handle message requests from JMS applications developed for WebSphere Application Server Version 5.1.

Before you begin

The WebSphere MQ client link is provided only for use with JMS applications developed for WebSphere Application Server Version 5.1.

About this task

The WebSphere MQ client link enables WebSphere Application Server Version 5.1 Java EE applications to use messaging resources of the default messaging provider of later versions that are backed by destinations on the service integration bus.

Each WebSphere MQ client link presents itself as a queue manager and transforms between the WebSphere MQ client protocols used by WebSphere Application Server Version 5.1[1] JMS applications.

If you migrate a Version 5.1 node to a later version, a default WebSphere MQ client link, with the name Default.MQClientLink, is created for the application servers on that node. You only have to create another WebSphere MQ client link if you want some Version 5.1 JMS applications to use a different channel to connect to the messaging engine, or want to present a messaging engine as a non-default queue manager name.

If you create a WebSphere MQ client link, you only have to manually specify values for a subset of the available properties. You can choose to specify values for other optional properties in this task, or if wanted at a later time by changing the configuration of the client link.

Tip: Creating a WebSphere MQ client link is only part of the task to enable WebSphere Application Server Version 5.1 applications to use messaging resources of the default messaging provider on a later version of the product. For more information about the overall task, see “Enabling WebSphere Application Server Version 5.1 JMS usage of messaging resources in later versions of the product” on page 583.

To create a WebSphere MQ client link, use the administrative console to complete the following steps.

Procedure

1. List the WebSphere MQ client links for the messaging engine that the Version 5.1 JMS applications are to connect to. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links**. This displays any existing WebSphere MQ client links in the content pane.
2. Click **New** in the content pane.
3. Specify the following required properties for the WebSphere MQ client link:

---

1. To make reading easier in this topic, the abbreviation “Version 5.1” is sometimes used to refer to “WebSphere Application Server Version 5.1”. For example, “Version 5.1 JMS resources” refers to JMS resources provided by WebSphere Application Server Version 5.1.

Name This can be any name that is useful for your administrative purposes. It is not used by the application environment.

MQ channel name
This is the name of the channel for the WebSphere MQ client link, used to flow messages between the JMS application developed for WebSphere Application Server Version 5.1 and the bus. This name must match the receiving channel name configured for WebSphere Application Server Version 5.1:

```
WAS.JMS.SVRCONN
```

This is the default value shown when you first display the WebSphere MQ client link settings panel.

Queue manager name
This is the virtual queue manager name that is associated with the messaging engine, and by which the messaging engine is known to WebSphere Application Server Version 5.1 applications. Type the queue manager name in the form:

```
WAS_node_name_server_name
```

Where:

node_name
is the name of the WebSphere Application Server node on the later version of the product.

server_name
is the name of the application server.

The correct value is shown by default when you first display the WebSphere MQ client link settings panel.

Default queue manager
Select this check box if you want the MQ client link to be used as the default for applications that cannot find a suitable MQ client link to use.

If a JMS application developed for WebSphere Application Server Version 5.1 specifies that it is to connect to a non-default queue manager name, you can configure a WebSphere MQ client link with that queue manager name. If a WebSphere MQ client link cannot be found with the required queue manager name, the connection is rejected. Alternatively, you can select this option on another WebSphere MQ client link, which is used instead of rejecting the connection.

4. Optional: If you want to specify other optional properties, see the property descriptions in “WebSphere MQ client link [Settings]” on page 2201.
5. Click **OK**.
6. Save your changes to the master configuration.

What to do next

To enable WebSphere Application Server Version 5.1 applications to use messaging resources of later versions, ensure you complete all the steps in the task “Enabling WebSphere Application Server Version 5.1 JMS usage of messaging resources in later versions of the product” on page 583.

Configuring a WebSphere MQ client link

Use this task to browse or change the properties of a client link that enables a messaging engine to handle message requests from JMS applications developed for WebSphere Application Server Version 5.1.

Before you begin

The WebSphere MQ client link is provided only for use with JMS applications developed for WebSphere Application Server Version 5.1.

About this task

The WebSphere MQ client link enables WebSphere Application Server Version 5.1 Java EE applications to use messaging resources of the default messaging provider in later versions that are backed by destinations on the service integration bus.

Each WebSphere MQ client link presents itself as a queue manager and transforms between the WebSphere MQ client protocols used by WebSphere Application Server Version 5.1[2] JMS applications.

Procedure

1. Start the administrative console.
2. Navigate to the settings page for the WebSphere MQ client link. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links -> *link_name***. This displays the properties of the selected WebSphere MQ client links in the content pane.
3. Optional: If you want to change any of the general properties, see the property descriptions in "WebSphere MQ client link [Settings]" on page 2201.
4. Optional: If you want to change any of the advanced properties, see the property descriptions in "WebSphere MQ client link advanced properties [Settings]" on page 2197, then complete the following steps.
5. If you have changed any properties, complete the following steps.
   a. Click **OK**.
   b. Save your changes to the master configuration.

Listing WebSphere MQ client links for a messaging engine

Use this task to display a list of WebSphere MQ links for a messaging engine.

About this task

To list the WebSphere MQ client links for a messaging engine, use the administrative console to complete the following steps.

Procedure

Display the WebSphere MQ client links collection page. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links**.

Results

This displays a list of WebSphere MQ client links for the messaging engine in the content pane, with the following subset of properties for the link:

Name The name of the WebSphere MQ client link.

Description
An optional description for the WebSphere MQ client link, for administrative purposes.

MQ channel name
The name of the channel for the WebSphere MQ client link, used to flow messages between WebSphere MQ clients and the bus.

---

2. To make reading easier in this topic, the abbreviation "Version 5.1" is sometimes used to refer to "WebSphere Application Server Version 5.1". For example, "Version 5.1 JMS resources" refers to JMS resources provided by WebSphere Application Server Version 5.1.

Queue manager name
The name of the WebSphere MQ queue manager on which the WebSphere MQ sender channel, that is connected to this MQ link receiver channel connection instance, is running.

Default queue manager
Whether or not this is the default queue manager for the WebSphere MQ clients.

Status
The runtime status of the WebSphere MQ client link.

What to do next

You can use this panel to create, delete, start, or stop client links, as described in related tasks.

Starting and stopping WebSphere MQ client links

Use this task to start or stop WebSphere MQ client links, to stop message flows between the default messaging provider and JMS applications developed for WebSphere Application Server Version 5.1, connected through the client links.

About this task

You can start a WebSphere MQ client link that is not running (has a runtime status of Stopped) or stop a client link that is running (has a runtime status of Started). This also starts or stops all client connections on the selected WebSphere MQ client links. Alternatively, you can start or stop individual client connections on the WebSphere MQ client link, as described in "Starting and stopping WebSphere MQ client connections" on page 592.

When stopping a client link, you can choose whether to force the client link to stop immediately or after any messages currently on its client connections have been processed. You can also choose the final state that you want the client link to be in when it has been stopped.

To start or stop one or more WebSphere MQ client links, use the administrative console to complete the following steps.

Procedure

1. Display the WebSphere MQ client links collection page. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links**.
2. Select the check box next to the name of each client link that you want to start or stop.
3. Click the appropriate button:

Start	Start selected items. This enables JMS applications to use the client link to access JMS resources provided by the default messaging provider.

Stop	Stop a selected WebSphere MQ client link. You must first have selected the link to be stopped. You can choose the mode of stop action and the required state when the link has been stopped: **Stop mode:** **Force** Stop the link immediately. **Quiesce** Stop the link after any messages currently on its client connections have been processed. **Target state:** **Inactive** Stop the link to an inactive state. If an application tries to use the link, the link is started again. **Stopped** Stop the link to a stopped state. Applications cannot use the client link. The link can only be started again by administrator action.

Results

The status of the client link changes and a message stating that the link has started or stopped is displayed at the top the page.

Listing client connections for a WebSphere MQ client link

Use this task to display a list of client connections for a WebSphere MQ client link. Each client connection is created automatically for a version 5 JMS application by using the WebSphere MQ client link.

About this task

To list the client connections for a WebSphere MQ client link, use the administrative console to complete the following steps.

Procedure

Display the client connections collection page. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links -> *link_name* [Additional Properties] Client connections**.

Results

This displays a list of client connections for the WebSphere MQ client link in the content pane, with the following subset of properties for the link:

IP address
The TCP/IP IP address of the WebSphere MQ client.

Status
The runtime status of the WebSphere MQ client connection.

What to do next

You can use this panel to start or stop client connections, as described in related tasks.

Starting and stopping WebSphere MQ client connections

Use this task to start or stop client connections on a WebSphere MQ client link, to stop message flows between the default messaging provider and the JMS application developed for WebSphere Application Server Version 5.1 that uses the connections.

About this task

You can start a WebSphere MQ client connection that is not running (has a runtime status of Inactive) or stop a client connection that is running (has a runtime status of Started). Alternatively, you can start or stop all client connections on the WebSphere MQ client link, as described in "Starting and stopping WebSphere MQ client links" on page 590.

When stopping a client connection, you can choose whether to force the client connection to stop immediately or after any messages currently on its client connections have been processed. Client connections are always stopped to an inactive state.

To start or stop one or more WebSphere MQ client connections, use the administrative console to complete the following steps.

Procedure

- Display the WebSphere MQ client connections collection page. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links -> *link_name* [Additional Properties] Client connections**.
- Select the check box next to the name of each client connection that you want to start or stop.
- Click the appropriate button:

Start	Start selected items.
Stop	Stop a selected WebSphere MQ client connection. You must first have selected the connection to be stopped. You can choose the mode of stop action, but connections are always stopped to an inactive state: **Stop mode:** **Force** Stop the connection immediately. **Quiesce** Stop the connection after any messages currently on the connection have been processed. **Target state:** **Inactive** Stop the link to an inactive state. If an application tries to use the link, the link is started again.

Results

The status of the client connection changes and a message stating that the connection has started or stopped is displayed at the top the page.

Deleting WebSphere MQ client links

Use this task to delete WebSphere MQ client links. This prevents JMS applications developed for WebSphere Application Server Version 5.1 from using those client links to exchange messages with the default messaging provider.

Before you begin

Before deleting a WebSphere Application Server Version 5 JMS applications using the client link.

- Stop all message-producing JMS applications in the WebSphere Application Server Version 5.1 environment that are using the client link. For example, you can use the administrative console to stop the applications, as described in Starting or stopping enterprise applications.
- Allow all message-consuming JMS applications (including those consuming publications as a result of durable subscriptions) to continue until all the JMS queues are drained, then stop those applications.

About this task

To start or stop one or more WebSphere MQ client links, use the administrative console to complete the following steps.

Procedure

1. Display the WebSphere MQ client links collection page. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links**.
2. Select the check box next to the name of each client link that you want to delete.
3. Click **Delete**
4. Click **OK**.
5. Save any changes to the master configuration.

Configuring the messaging engine selection process for JMS applications

Configure the JMS connection factory for your application, in order to tune the process through which messaging engine connections are selected for your application.

About this task

To use JMS destinations of the default messaging provider, a client application connects to a messaging engine on the service integration bus to which the destinations are assigned. For example, a JMS queue is assigned to a queue destination on a service integration bus.

By default, the environment automatically connects applications to an available messaging engine on the bus. However you can specify extra configuration details to influence the connection process; for example to identify special bootstrap servers, or to limit connection to a subgroup of available messaging engines, or to improve availability or performance, or to ensure sequential processing of messages received.

For a JMS application, you apply the extra configuration to the associated JMS connection factory. For a message-driven bean (MDB) application, you apply the equivalent extra configuration to the associated activation specification.

For the default configuration, you only have to specify the one required connection property **Bus name**, which sets the name of the bus to which the application is to connect. To further restrict the range of messaging engines to which your applications can connect, you can also configure the other connection properties:

- Target
- Target type
- Target significance
- Target inbound transport chain
- Connection proximity

For detailed information about these connection properties, and an overview of the process through which the default messaging provider chooses the messaging engine for your application, see How JMS applications connect to a messaging engine on a bus.

The steps for this task are based on an application that uses a unified JMS connection factory. You can use the same task to configure a JMS queue connection factory or JMS topic connection factory, but you select the appropriate type of connection factory instead of **JMS connection factory**.

Procedure

- If the client application uses a JMS connection factory in the client container, use the Client Resource Configuration tool (ACRCT) to configure the Provider endpoint property:
  1. Start the tool and open the EAR file for which you want to configure the JMS connection factory. The EAR file contents are displayed in a tree view.
  2. From the tree, select the JAR file in which you want to configure the JMS connection factory.
  3. Expand the JAR file to view its contents.
  4. Expand **Messaging Providers > Default Provider > Connection Factories**.
  5. Optional: Display the general properties of the connection factory:
     - If you want to use an existing JMS connection factory, click the name of the connection factory.
     - If you want to create a new JMS connection factory, click **New**.

       For more information about configuring a JMS connection factory in the JMS provider configuration for your application client, see “Configuring Java messaging client resources” on page 72.
  6. On the General tab, configure the connection properties.
  7. Click **OK**.
  8. To save your changes, click **File > Save**.
- If the client application uses a JMS connection factory on the server, use the WebSphere Application Server administrative console to configure the connection properties:
  1. Display the default messaging provider. In the navigation pane, expand **Resources -> JMS -> JMS providers**.
  2. Optional: Change the **Scope** check box to set the level at which the connection factory is to be visible, according to your needs.
  3. In the content pane click **Default messaging provider**. This displays a table of properties for the default messaging provider, including links to the types of JMS resources that it provides.
  4. In the content pane, under Additional Properties, click **Connection factories** This displays any existing connection factories in the content pane.
  5. Optional: Display the general properties of the connection factory:
     - If you want to use an existing JMS connection factory, click the name of the connection factory.
     - If you want to create a new JMS connection factory, click **New**.

For more information about configuring a JMS connection factory, see “Configuring a unified connection factory for the default messaging provider” on page 518.

6. Configure the connection properties.
7. Click **OK**.
8. Save your changes to the master configuration.

Managing messages and subscriptions for default messaging JMS destinations

You can manage the messages and subscriptions that exist for JMS destinations of the default messaging provider. You can manage the messages on a JMS queue by acting on the queue point for the bus destination to which JMS queue has been assigned. You can administer the durable subscriptions on a JMS topic by acting on a publication point for the topic space to which JMS topic has been assigned.

About this task

The **Bus name** property of the JMS connection factory identifies the service integration bus. For managing the messages on a JMS queue, note that the **Queue name** property of the JMS queue identifies the name of the bus destination. For administering the durable subscriptions on a JMS topic, note that the **Topic space** property of the JMS topic identifies the name of the topic space.

Queue points and publication points are examples of message points. For information about how to manage messages on message points, see the following subtopics.

Procedure

- Manage messages on message points.
- Administer durable subscriptions.

Managing messages on message points

Use these tasks to list and act on runtime messages that exist on message points in a service integration bus.

About this task

You can list the message points for bus destinations and messaging engines, and list the messages on a selected message point. You can use the list of messages as part of a troubleshooting task to find messages that need to be deleted.

- “Listing messages on a message point”
- “Deleting messages on a message point” on page 596

Listing messages on a message point:

Use this task to list the messages that exist on a message point for a selected bus destination or messaging engine.

About this task

To display a list of messages on a message point, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the service integration bus.
3. Optional: To list the message points for a bus destination, complete the following steps:

a. In the content pane, under **Destination resources**, click **Destinations**.
b. Click the destination name.

4. Optional: To list the message points for a messaging engine, complete the following steps:
   a. In the content pane, under **Topology**, click **Messaging engines**.
   b. Click the messaging engine name.
5. Under Additional Properties, click **Message points**. This displays a list of message points in the content pane.
6. Click the message point name. This displays the properties of the destination localization in the content pane.
7. Click the Runtime tab.
8. Under Additional Properties, click **Messages**.

Results

A list of messages on the selected message point is displayed in the content pane.

What to do next

You can select one or more messages to act on; for example, to display the message content, delete messages.

Deleting messages on a message point:

Use this task to delete one or more messages that exist on a message point for a selected bus destination or messaging engine.

About this task

You should not usually have to delete messages on a message point. This task is intended as part of a troubleshooting procedure.

To delete one or messages on a message point, use the administrative console to complete the following steps:

Procedure

1. List the messages on the message point.
2. In the content pane, select the check box next to each message that you want to delete. Alternatively, you can select all messages in the list by clicking **Select all items**  .
3. Click **Delete**.

Results

The selected messages are removed from the list.

Using JMS from stand-alone clients to interoperate with service integration resources

The Thin Client for JMS with WebSphere Application Server allows third party applications to interoperate with default messaging provider messaging engines on WebSphere Application Server.

Using JMS to connect to a WebSphere Application Server default messaging provider messaging engine

The Thin Client for JMS with WebSphere Application Server is an embeddable technology that provides Java Message Service (JMS) V1.1 connections to a WebSphere Application Server default messaging provider messaging engine.

Installing and configuring the Thin Client for JMS with WebSphere Application Server:

To use the Thin Client for JMS with WebSphere Application Server copy the com.ibm.ws.sib.client.thin.jms_7.0.0.jar and any other required files from the application server or application client %WAS_HOME%/runtimes directory.

About this task

The Thin Client for JMS with WebSphere Application Server can be used for default messaging provider messaging engines for WebSphere Application Server Version 6.0.2 or later. The connection to the messaging engine can be either TCP or SSL. HTTP connectivity is not supported.

Installation and configuration of the client in an OSGi environment is different and described in “Installing and configuring the Thin Client for JMS with WebSphere Application Server in an OSGi environment” on page 599.

You can install the client in any location and run it in any supported Java 2 Platform Standard Edition 1.5.0 (also known as 5.0) or above Java Runtime Environment (JRE). The client supports the following JREs:

- IBM JRE 1.5.0 and above
- Sun JRE 1.5.0 and above
- HP-UX JRE 1.5.0 and above
- Lotus® Expeditor Version 6.1 or above with J2SE 5.0 or above Device Runtime Environment. jclDesktop and jclDevice profiles are not supported.

The client does not require any further configuration after installation, apart from adding the jar file or files to the classpaths for your client application. You can choose either to create JMS connection factories programmatically, or use the Java Naming and Directory Interface (JNDI). If required, you can use secure connections by configuring Secure Sockets Layer (SSL) settings.

Procedure

1. Install the client in the required location. The client is always installed in the `/runtimes` directory of a WebSphere Application Server installation, and might optionally be installed by the Application Client for WebSphere Application Server, which is a separate WebSphere Application Server deliverable. The client is shipped as three files:
   - `com.ibm.ws.sib.client.thin.jms_7.0.0.jar` - the regular JMS Client.
   - `com.ibm.ws.sib.client_ExpeditorDRE_7.0.0.jar` - the JMS Client packaged for Lotus Expeditor.
   - `sibc.nls.zip` - language-specific resource bundles. You can extract any combination of these files. The client jar already includes US English, so you only need the additional language files from `sibc.nls.zip` if you require languages other than non-US English.
2. Include the appropriate jar file or files in the classpaths for your client application:
   a. To compile JMS code, include the client jar file in the CLASSPATH setting for the javac command.
   b. To run JMS code, include the client jar file and any required optional language files extracted from `sibx.nls.zip` in the CLASSPATH setting for the java command.
3. Configure the required JMS resources as described in “Using JMS resources with the Thin Client for JMS with WebSphere Application Server” on page 599.
4. If you require secure connections, configure SSL as described in “Securing JMS client and JMS resource adapter connections” on page 601.

Migration to the Thin Client for JMS with WebSphere Application Server:

There are a number of differences to consider when migrating to the Thin Client for JMS with WebSphere Application Server from an earlier version of the client.

Table 51. Migration from WebSphere Application Server Version 6.0.2 to WebSphere Application Server Version 7.0 or later. The first column of the table lists the areas where the differences are found between WebSphere Application Server Version 6.0.2 and WebSphere Application Server Version 7.0 or later. The second column lists the availability, applicability, file path, JRE level or the property names of the specific area of WebSphere Application Server Version 6.0.2 (Client for Java Message Service on Java 2 Platform, Standard Edition with WebSphere Application Server). The third column provides the file path, file names, JRE level or property names of the specific area of WebSphere Application Server Version 7.0 or later (Thin Client for JMS with WebSphere Application Server).

Area	**WebSphere Application Server Version 6.0.2 (Client for Java Message Service on Java 2 Platform, Standard Edition with WebSphere Application Server)**	**WebSphere Application Server Version 7.0 or later (Thin Client for JMS with WebSphere Application Server)**
Installation	Available for separate download and installation	Installed in the WebSphere Application Server or Application Client /runtimes directory
JMS jar file name	`sibc.jms.jar` file	`com.ibm.ws.sib.client.thin.jms_`*`version_number`*`.jar`
Performing JNDI lookups of JMS resources	Requires optional `sibc.jndi.jar` file (and ORB `sibc.orb.jar` file for non-IBM JREs)	Requires Thin Client for EJB with WebSphere Application Server jar file `com.ibm.ws.ejb.thinclient_`*`version_number`*`.jar` or `com.ibm.ws.ejb.thinclient.z_`*`version_number`*`.jar` (and ORB `com.ibm.ws.orb_`*`version_number`*`.jar` for non-IBM JREs)
SSL configuration	Secure connections are configured by using JRE global properties: **-Djavax.net.ssl.keyStore** **-Djavax.net.ssl.keyStorePassword** **-Djavax.net.ssl.trustStore** **-Djavax.net.ssl.trustStorePassword**	Two approaches to configuring secure connections. The first approach uses JRE global properties: **-Djavax.net.ssl.keyStore** **-Djavax.net.ssl.keyStorePassword** **-Djavax.net.ssl.trustStore** **-Djavax.net.ssl.trustStorePassword** The second approach is to specify security settings that are specific to Thin Client for JMS with WebSphere Application Server connections: **-Dcom.ibm.ssl.keyStoreType** **-Dcom.ibm.ssl.keyStore** **-Dcom.ibm.ssl.keyManager** **-Dcom.ibm.ssl.trustManager** **-Dcom.ibm.ssl.keyStorePassword** **-Dcom.ibm.ssl.protocol** **-Dcom.ibm.ssl.contextProvider** **-Dcom.ibm.ws.sib.jsseProvider**
Minimum JRE level	1.4.2	1.5
Enable trace (set trace specification)	**-Dcom.ibm.ws.sib.client.traceSetting**	**-Dcom.ibm.ejs.ras.lite.traceSpecification**
Set trace file name	**-Dcom.ibm.ws.sib.client.traceFile**	**-Dcom.ibm.ejs.ras.lite.traceFileName**
Set max trace file size	Not applicable	**-Dcom.ibm.ejs.ras.lite.maxFileSize**
Set max number of trace files	Not applicable	**-Dcom.ibm.ejs.ras.lite.maxFiles**
Set trace format	Not applicable	**-Dcom.ibm.ejs.ras.lite.traceFormat**

Table 51. Migration from WebSphere Application Server Version 6.0.2 to WebSphere Application Server Version 7.0 or later (continued). The first column of the table lists the areas where the differences are found between WebSphere Application Server Version 6.0.2 and WebSphere Application Server Version 7.0 or later. The second column lists the availability, applicability, file path, JRE level or the property names of the specific area of WebSphere Application Server Version 6.0.2 (Client for Java Message Service on Java 2 Platform, Standard Edition with WebSphere Application Server). The third column provides the file path, file names, JRE level or property names of the specific area of WebSphere Application Server Version 7.0 or later (Thin Client for JMS with WebSphere Application Server).

Area	WebSphere Application Server Version 6.0.2 (Client for Java Message Service on Java 2 Platform, Standard Edition with WebSphere Application Server)	WebSphere Application Server Version 7.0 or later (Thin Client for JMS with WebSphere Application Server)
Using alternative trace properties file	**-DtraceSettingsFile**	**-DtraceSettingsFile** Thin Client for JMS with WebSphere Application Server supports new options described in "Trace user interface for stand-alone clients" on page 603
Enable and specify FFDC file name	**-DffdcLogFile**	**-Dcom.ibm.ejs.ras.lite.ffdcLogFile**
NLS support	Installation option for NLS support	Non-English versions are available in the `sibc.nls.jar` file

Installing and configuring the Thin Client for JMS with WebSphere Application Server in an OSGi environment:

Use this information to install and configure the Thin Client for JMS with WebSphere Application Server in an OSGi environment.

About this task

To use the Thin Client for JMS with WebSphere Application Server in an OSGi environment import the required plug-ins into your development environment and add them to the list of required plug-ins in your product configuration.

A list of all packages exported by system.bundle can be obtained from the Execution Environment Profile being used. For example if the Execution Environment is J2SE-1.5 then the list of all packages exported by system.bundle can be obtained from the property org.osgi.framework.system.packages in the Execution Environment Profile file named J2SE-1.5.profile contained in the bundle org.eclipse.osgi_<version>.jar

Procedure

- Application plug-ins which use the Thin Client for JMS with WebSphere Application Server plug-ins must import the javax.jms packages in an OSGi environment.
- If you are using connection factories that are programmatically created, the application plug-in must import the com.ibm.websphere.sib.api.jms package.
- Alternatively, if you are using JNDI lookups via the Thin Client for EJB in WebSphere Application server then the application plug-in must import the com.ibm.websphere.naming package.
- When performing JNDI lookups it is necessary that the following system property be specified:
  ```
  -Dorg.osgi.framework.system.packages=sun.io,com.ibm.wsspi.channel.framework,
  com.ibm.CORBA.channel,com.ibm.CORBA.channel.giop,com.ibm.channel.orb,
  com.ibm.CORBA.poa,com.ibm.CORBA.ras,com.ibm.CORBA.iiop,com.ibm.CORBA.transport
  ```

Using JMS resources with the Thin Client for JMS with WebSphere Application Server:

Suitable JMS connection factories and references to JMS queues or topics might be obtained programmatically without using JNDI. Alternatively, full JNDI support might be obtained from the Thin Client for EJB with WebSphere Application Server.

Procedure

- To obtain suitable connection factories programmatically, without using JNDI, use code similar to that shown in the following example:

```
import com.ibm.websphere.sib.api.jms.*;
...
JmsConnectionFactory jmsCF =
  JmsFactoryFactory.getInstance().createQueueConnectionFactory();
jmsCF.setBusName("myBus");
jmsCF.setProviderEndpoints("1.2.3.4");
```

  To obtain a suitable reference to a JMS queue or topic programmatically, use code similar to that shown in the following example:

```
JmsQueue jmsQ = JmsFactoryFactory.getInstance().createQueue("myQueue");
```

  For further information, see the JmsFactoryFactory class API documentation available with WebSphere Application Server.

- To obtain full JNDI support from the Thin Client for EJB with WebSphere Application Server:
  1. Include the `/runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar` file in the compile and runtime classpaths for your enterprise application as described in “Installing and configuring the Thin Client for JMS with WebSphere Application Server” on page 597.
  2. Use the following code to create a suitable Initial Context, substituting the server IP address and port as appropriate:

```
import javax.naming.*;
...
Properties env = new Properties();
env.put(Context.PROVIDER_URL,"iiop:
  //<server IP address>:<server bootstrap address port>");
env.put(Context.INITIAL_CONTEXT_FACTORY,
  "com.ibm.websphere.naming.WsnInitialContextFactory");
InitialContext ctx = new InitialContext(env);
```

     In certain situations, for example when running with a Sun JRE, an additional ORB jar is also required. For additional information about when this jar is required, see “Running the IBM Thin Client for Enterprise JavaBeans (EJB)” on page 155.

Obtaining WebSphere MQ JMS resources in the thin client environment:

A stand-alone Java SE JMS thin client application that connects to an external WebSphere MQ queue manager can get administratively-created WebSphere MQ messaging provider JMS resources from the WebSphere Application Server Java Naming and Directory Interface (JNDI) namespace.

Procedure

1. To obtain WebSphere MQ messaging provider JMS resources from the WebSphere Application Server JNDI namespace in the thin client environment, include the following jar files in the runtime classpath of your application:
   - A copy of the `/runtimes/com.ibm.ws.ejb.thinclient_7.0.0.jar` file.
   - A copy of the `/runtimes/com.ibm.ws.messagingClient.jar` file.
   - WebSphere MQ client jar files, which must be obtained from the WebSphere MQ product.
2. Use the following code to create a suitable Initial Context, substituting the server IP address and port as appropriate:

```
import javax.naming.*;
...
Properties env = new Properties();
env.put(Context.PROVIDER_URL,"iiop:
  //<server IP address>:<server bootstrap address port>");
env.put(Context.INITIAL_CONTEXT_FACTORY,
  "com.ibm.websphere.naming.WsnInitialContextFactory");
InitialContext ctx = new InitialContext(env);
```

In certain situations, for example when running with a Sun JRE, an additional ORB jar is also required. For additional information about when this jar is required, see the Thin Client for EJB with WebSphere Application Server information.

Securing JMS client and JMS resource adapter connections

There are two approaches to configuring Secure Sockets Layer (SSL) for the Thin Client for JMS with WebSphere Application Server and the Resource Adapter for JMS with WebSphere Application Server. The global configuration approach affects all stand-alone outbound connections from the process, and the private approach applies only to client or resource adapter connections from the process.

About this task

The Thin Client for JMS with WebSphere Application Server and the Resource Adapter for JMS with WebSphere Application Server use the standard Java Secure Socket Extension (JSSE) that all supported JREs provide for making Secure Sockets Layer (SSL) connections. For information about JSSE, see the JSSE documentation.

The global configuration approach uses JRE global properties and affects all outbound SSL connections that your application initiates. For a JRE configured to use SSL connections to connect to WebSphere Application Server, you typically have to set the following javax.net.ssl system properties:

```
-Djavax.net.ssl.keyStore=key.p12
-Djavax.net.ssl.keyStorePassword={xor}Lz4sLCgwLTs=
-Djavax.net.ssl.trustStore=trust.p12
-Djavax.net.ssl.trustStorePassword={xor}PSo4LSov
```

You can use the private configuration approach to specify security settings that are specific to the Thin Client for JMS with WebSphere Application Server or the Resource Adapter for JMS with WebSphere Application Server connections. You can configure the com.ibm.ws.sib.client.ssl.properties system property to specify the location of an IBM SSL properties file. If this system property is not configured, an attempt is made load the properties file from the classpath instead.

The client obtains the value that it uses for any particular SSL property as follows:

- If the property has a value defined in the properties file containing the IBM SSL properties, the client uses this value.
- If there is no value for the property in the properties file, and there is a suitable property in the associated JRE system properties, the client uses this value.
- If there is no suitable javax.net.ssl property, the client uses the default value.

The table below summarizes the IBM SSL property keys that can be configured inside the IBM SSL properties file, and the corresponding javax.net.ssl.* system property keys and default values.

Table 52. IBM SSL property values and corresponding JRE global property and default values. The first column of the table lists the IBM SSL property keys and the second column lists the corresponding JRE global property keys. The third column provides the default values of the properties.

IBM SSL property	JRE global property	Default value
com.ibm.ssl.keyStoreType	javax.net.ssl.keyStoreType	JKS
com.ibm.ssl.keyStore	javax.net.ssl.keyStore	None

Table 52. IBM SSL property values and corresponding JRE global property and default values (continued). The first column of the table lists the IBM SSL property keys and the second column lists the corresponding JRE global property keys. The third column provides the default values of the properties.

IBM SSL property	JRE global property	Default value
com.ibm.ssl.keyManager	javax.net.ssl.keyStoreProvider	IbmX509
com.ibm.ssl.trustManager	javax.net.ssl.trustStoreProvider	IbmX509
com.ibm.ssl.keyStorePassword	javax.net.ssl.keyStorePassword	None
com.ibm.ssl.protocol	None	SSL
com.ibm.ssl.contextProvider	None	IBMJSSE2
com.ibm.ws.sib.jsseProvider	None	com.ibm.jsse2.IBMJSSEProvider2
com.ibm.ssl.trustStore	javax.net.ssl.trustStore	None
com.ibm.ssl.trustStoreType	javax.net.ssl.trustStoreType	JKS
com.ibm.ssl.trustStorePassword	javax.net.ssl.trustStorePassword	None

For example, you might create an ssl.properties file that contains the following properties and values:

```
com.ibm.ssl.keyStore=/thinclient/key.p12
com.ibm.ssl.keyStoreType=PKCS12
com.ibm.ssl.keyStorePassword=WebAS
com.ibm.ssl.trustStore=/thinclient/trust.p12
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStorePassword=WebAS
```

You can use the PropFilePasswordEncoder tool in the WebSphere Application Server bin directory to encode passwords stored in plain text property files. For further information see Encoding passwords in files.

Notes:

1. SSL connections from SUN JREs that use the Thin Client for JMS with WebSphere Application Server cannot use the default WebSphere Application Server PKCS12 key and trust stores. If you are running the client securely from SUN JREs, you must first extract the certificates from the trust store by using an IBM software development kit (SDK). You can then import these certificates into a keystore that the Sun JRE can recognize correctly, such as a JKS keystore.
2. SSL connections are not supported by the IBM JRE shipped with WebSphere Application Server - a non-WebSphere Application Server installed JRE must be used.

Procedure

1. Obtain the necessary key and trust store files.
2. Set the javax.net.ssl system properties required for the global configuration approach.
3. For the private configuration approach, use the com.ibm.ws.sib.client.ssl.properties system property to specify the file from which the SSL properties are to be loaded, as shown in the following example:

   ```
   -Dcom.ibm.ws.sib.client.ssl.properties=c:/ssl.properties
   ```

Adding tracing and logging for stand-alone clients

You can add tracing and logging to help analyze performance and diagnose problems.

About this task

This information applies to the following WebSphere Application Server stand-alone clients:

- Thin Client for JMS with WebSphere Application Server
- Thin Client for EJB with WebSphere Application Server
- Thin Client for JAX-WS with WebSphere Application Server

- Thin Client for JAX-RPC with WebSphere Application Server

Procedure

- To enable trace, use either a long form or short form system property.

  Note: Trace settings are determined from the system property values the first time that a WebSphere Application Server client is called. The trace settings are then fixed. Therefore, any subsequent changes to the system property settings do not change the trace settings that the WebSphere Application Server client uses.
- To enable First Failure Data Capture (FFDC), use either a long or short form system property.

  Note: FFDC settings are determined from the system property values the first time that a WebSphere Application Server client performs an FFDC. The FFDC settings are then fixed. Therefore, any subsequent changes to the system property settings do not change the FFDC settings that the WebSphere Application Server client uses.

Trace user interface for stand-alone clients:

To enable trace, you can either use a long form or a short form system property.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Long form system properties

The long form system property takes priority over the short form and uses system properties that are unique to WebSphere Application Server.

Table 53. Long form system properties. The table contains the list of long form system properties and the descriptions of the properties.

Property	Description
com.ibm.ejs.ras.lite.traceSpecification	The trace specification string
com.ibm.ejs.ras.lite.traceFileName	The trace destination (<*file*>, stdout, stderr, java,util.logging)
com.ibm.ejs.ras.lite.maxFileSize	The maximum trace file size in MB (if the trace destination is a file)
com.ibm.ejs.ras.lite.maxFiles	The maximum number of trace files kept (if the trace destination is a file)
com.ibm.ejs.ras.lite.traceFormat	The trace output format, which can be either basic or advanced (the default is basic)

Long form example:

```
-Dcom.ibm.ejs.ras.lite.traceSpecification=SIB*=all
-Dcom.ibm.ejs.ras.lite.traceFileName=c:/trace.log
-Dcom.ibm.ejs.ras.lite.maxFileSize=20
-Dcom.ibm.ejs.ras.lite.maxFiles=8
```

Short form system property

The short form uses a system property that is compatible with existing WebSphere Application Server clients but that, because this property is unqualified, might clash with other third party technologies that are running in the same Java runtime environment (JRE).

The short form system property is:

```
traceSettingsFile
```

This property must specify a loadable properties file that can contain the following properties:

Table 54. Properties in loadable system properties file. The table includes the list of loadable system properties and the descriptions of the properties.

Property	Description
traceFileName	The trace destination (file, stdout, stderr, java,util.logging)
maxFilesSize	The maximum trace file size in MB (if the trace destination is a file)
maxFiles	The maximum number of trace files kept (if the trace destination is a file)
<traceSpec>	The trace specification
traceFormat	The trace output format, which can be either basic or advanced (the default is basic)

The following example shows how to use the short form system property:

```
SIBTrm=all:SIBMfp=all
traceFileName=c:/trace.log
```

Special meanings for trace file name values

Some trace file name values have a special meaning:

- stdout - causes trace records to be written to stdout
- stderr - causes trace records to be written to stderr
- java.util.logging - causes trace records to be written to java.util.logging

Using any other name causes the trace records to be written to a file of that name.

First Failure Data Capture user interface for stand-alone clients:

To enable First Failure Data Capture (FFDC) output, you can either use a long or short form system property.

Long form system property

The long form takes priority over the short form and uses a system property that is unique to WebSphere Application Server to specify the FFDC dump file name. This property is:

```
com.ibm.ejs.ras.lite.ffdcLogFile
```

The following example shows how to use the long form system property to enable FFDC output:

```
-Dcom.ibm.ejs.ras.lite.ffdcLogFile=c:\ffdc.log
```

Short form system property

The short form system property is:

```
ffdcLogFile
```

The following example shows how to use the short form system property to enable FFDC output:

```
-DffdcLogFile=c:\ffdc.log
```

Using JMS from a third party application server to interoperate with service integration resources

The Resource Adapter for JMS with WebSphere Application Server provides first class connectivity to service integration resources from the third party application server on which it is deployed.

About this task

The Resource Adapter for JMS with WebSphere Application Server is designed to be deployed into third party application servers, that is, into non-WebSphere Application Server application servers, that support Java EE Connector Architecture (JCA) 1. 5 and are Java 2 Platform, Enterprise Edition (J2EE) 1.4 compliant. The stand-alone resource adapter provides these third party application servers with full connectivity to service integration resources running inside WebSphere Application Server Version 6.0.2 and later.

Supported third party application servers are:

- WebSphere Application Server Community Edition Version 2.0 and later
- Apache Geronimo v2.0 and later
- JBoss Application Server v4.0.5 and later

Restriction:

- The resource adapter does not support the unmanaged JCA environment. In an unmanaged environment use the Thin Client for JMS with WebSphere Application Server.
- If you want to use XA connections to a WebSphere Application Server Version 6.0.2 application server, contact IBM Support to obtain a required service update.

Deploying the Resource Adapter for JMS with WebSphere Application Server to a third party application server

To provide connections to service integration resources running inside WebSphere Application Server, the Resource Adapter for JMS with WebSphere Application Server must be installed into the third party application server.

Before you begin

The Resource Adapter for JMS with WebSphere Application Server requires JRE 1.5 or later. The resource adapter is called sibc.jmsra.rar and is available from the following runtime directories:

- WebSphere Application Server install
- Application Client for WebSphere Application Server install

Before starting the deployment of the resource adapter, you must first obtain the following information from the WebSphere Application Server administrator:

- Bus name
- Endpoint provider address
- Target transport chain
- Messaging engine name

- Any other required connection and destination properties
- One or more destination names

The general approach to deploying the resource adapter is to write a deployment XML file to configure the required and optional properties for the JMS connection factory and JMS resources that will be accessed, and then deploy the resource adapter by using the deployment XML file. The installation process varies, depending on the particular application server that you are using. Before starting this task, see the documentation specific to your application server for information about how to install and use a JMS resource adapter RAR file.

About this task

An enterprise application that looks up a Resource Adapter for JMS with WebSphere Application Server connection factory in the local Java Naming and Directory Interface (JNDI) repository can access service integration resources through the resource adapter, provided that the required messaging engine is available in WebSphere Application Server. All outbound connections must access all queues and topics by using Queue or Topic resources. These resources are configured using your particular application server configuration mechanism when the resource adapter is deployed.

The Resource Adapter for JMS with WebSphere Application Server supports full two-phase XA transactional connections (except under the JBoss Application Server) but can also be run using local transactions or no transaction connections.

Multiple deployments of the resource adapter are possible.

Procedure

1. To deploy an outbound JMS resource on the Resource Adapter for JMS with WebSphere Application Server, use your particular application server configuration mechanism to configure the following service integration bus properties:
   - Bus name
   - Provider endpoints
2. If you want to use XA resources over a Resource Adapter for JMS with WebSphere Application Server connection, use your particular application server configuration mechanism to configure the following additional service integration bus properties:
   - Target type must be set to "ME"
   - Target significance must be set to "Required"
   - Target must be set to the name of the required messaging engine

   These properties permit the recovery of indoubt transactions, should this be necessary. For further information about indoubt transactions, see ../ae/tjm0165_.dita.

   See “Configuration properties for the Resource Adapter for JMS with WebSphere Application Server” on page 607 for a description of these property names and other properties that might also be configured.

Results

Subsequent usage of the resource adapter is in accordance with the Java EE programming specifications. That is, any enterprise bean or message-driven bean might obtain a Resource Adapter for JMS with WebSphere Application Server connection factory or use an activation specification to connect to a service integration messaging engine. Message-driven beans behave in just the same way as they would in any other Java EE environment.

What to do next

You can turn trace and First Failure Data Capture (FFDC) on for the resource adapter in the same way as for the Thin Client for JMS with WebSphere Application Server. For further information, see “Adding tracing and logging for stand-alone clients” on page 602.

You can configure secure connections by configuring connection factories that require a secure bootstrap and/or connection transport chain in the same way as for the Thin Client for JMS with WebSphere Application Server. For further information, see “Securing JMS client and JMS resource adapter connections” on page 601.

Configuration properties for the Resource Adapter for JMS with WebSphere Application Server:

As part of deploying the Resource Adapter for JMS with WebSphere Application Server, you must configure a set of JMS resources that the deployed resource adapter instance supports.

The following tables list the JMS properties and their values.

Note: Not all properties available to applications running inside the WebSphere Application Server environment are available in third party environments. Some properties have no meaning outside of the WebSphere Application Server environment and some have no meaning for remotely connected clients.

Table 55. Connection factory properties. The first column of the table lists the connection factory property names. The second column provides the description of the properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
BusName	The name of the service integration bus to connect to.		
ClientID	The JMS client identifier needed for durable topic subscriptions on all connections created using this connection factory.		
UserName			
Password			
NonPersistentMapping	The reliability applied to nonpersistent JMS messages sent using this connection factory.	BestEffortNonPersistent, ExpressNonPersistent, ReliableNonPersistent	ExpressNonPersistent
PersistentMapping	The reliability applied to persistent JMS messages sent using this connection factory.	ReliablePersistent, AssuredPersistent	ReliablePersistent
DurableSubscriptionHome	The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS connection factory.		
ReadAhead	Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.	AlwaysOn, AlwaysOff, Default	Default

Table 55. Connection factory properties (continued). The first column of the table lists the connection factory property names. The second column provides the description of the properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
Target	The name of a target that identifies a group of messaging engines. Specify the type of target by using the Target type property.		
TargetType	The type of target named in the Target property.	BusMember, Custom, ME	BusMember
TargetSignificance	The significance of the target group.	Required, Preferred	Required
TargetTransportChain	The name of the protocol that resolves to a group of messaging engines.		
ProviderEndpoints	The list of comma separated endpoints used to connect to a bootstrap server.		
ConnectionProximity	The proximity of messaging engines that can accept connection requests, in relation to the bootstrap messaging engine.	Server, Cluster, Host, Bus	Bus
TemporaryQueueNamePrefix	The prefix of up to twelve characters used for names of temporary queues created by applications that use this connection factory.		
TemporaryTopicNamePrefix	The prefix used at the start of temporary topics created by applications that use this connection factory.		
ShareDurableSubscriptions	Controls whether durable subscriptions are shared across connections with members of a server cluster.	InCluster, AlwaysShared, NeverShared	InCluster (always resolves to AlwaysOff as the client is always outside of a WebSphere Application Server clustered server)
ProducerDoesNotModify PayloadAfterSet	When enabled, Object or Bytes messages sent by a message producing application that has connected to the bus by using this connection factory will not have their data copied when set and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.	true, false	false

Table 56. Queue Connection factory properties. The first column of the table lists the queue connection factory property names. The second column provides the description of the properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
BusName	The name of the service integration bus to connect to.		

Table 56. Queue Connection factory properties (continued). The first column of the table lists the queue connection factory property names. The second column provides the description of the properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
UserName			
Password			
NonPersistentMapping	The reliability applied to nonpersistent JMS messages sent using this connection factory.	BestEffortNonPersistent, ExpressNonPersistent, ReliableNonPersistent	ExpressNonPersistent
PersistentMapping	The reliability applied to persistent JMS messages sent using this connection factory.	ReliablePersistent, AssuredPersistent	ReliablePersistent
ReadAhead	Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.	AlwaysOn, AlwaysOff, Default	Default
Target	The name of a target that identifies a group of messaging engines. Specify the type of target by using the Target type property.		
TargetType	The type of target named in the Target property.	BusMember, Custom, ME	BusMember
TargetSignificance	The significance of the target group.	Required, Preferred	Required
TargetTransportChain	The name of the protocol that resolves to a group of messaging engines.		
ProviderEndpoints	The list of comma separated endpoints used to connect to a bootstrap server.		
ConnectionProximity	The proximity of messaging engines that can accept connection requests, in relation to the bootstrap messaging engine.	Server, Cluster, Host, Bus	Bus
TemporaryQueueNamePrefix	The prefix of up to twelve characters used for names of temporary queues created by applications that use this connection factory.		
ProducerDoesNotModify PayloadAfterSet	When enabled, Object or Bytes messages sent by a message producing application that has connected to the bus by using this connection factory will not have their data copied when set and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.	true, false	false

Table 57. Topic Connection factory properties. The first column of the table lists the topic connection factory property names. The second column provides the description of the topic connection factory properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
BusName	The name of the service integration bus to connect to.		
ClientID	The JMS client identifier needed for durable topic subscriptions on all connections created by using this connection factory.		
UserName			
Password			
NonPersistentMapping	The reliability applied to nonpersistent JMS messages sent using this connection factory.	BestEffortNonPersistent, ExpressNonPersistent, ReliableNonPersistent	ExpressNonPersistent
PersistentMapping	The reliability applied to persistent JMS messages sent using this connection factory.	ReliablePersistent, AssuredPersistent	ReliablePersistent
DurableSubscriptionHome	The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS connection factory.		
ReadAhead	Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.	AlwaysOn, AlwaysOff, Default	Default
Target	The name of a target that identifies a group of messaging engines. Specify the type of target by using the Target type property.		
TargetType	The type of target named in the Target property.	BusMember, Custom, ME	BusMember
TargetSignificance	The significance of the target group.	Required, Preferred	Required
TargetTransportChain	The name of the protocol that resolves to a group of messaging engines.		
ProviderEndpoints	The list of comma separated endpoints used to connect to a bootstrap server.		
ConnectionProximity	The proximity of messaging engines that can accept connection requests, in relation to the bootstrap messaging engine.	Server, Cluster, Host, Bus	Bus
TemporaryTopicNamePrefix	The prefix used at the start of temporary topics created by applications that use this connection factory.		

Table 57. Topic Connection factory properties (continued). The first column of the table lists the topic connection factory property names. The second column provides the description of the topic connection factory properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
ShareDurableSubscriptions	Controls whether durable subscriptions are shared across connections with members of a server cluster.	InCluster, AlwaysShared, NeverShared	InCluster (always resolves to AlwaysOff as the client is always outside of a WebSphere Application Server clustered server)
ProducerDoesNotModify PayloadAfterSet	When enabled, Object or Bytes messages sent by a message producing application that has connected to the bus by using this connection factory will not have their data copied when set and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.	true, false	false

Table 58. Queue properties. The first column of the table lists the queue property names. The second column provides the description of the queue properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
QueueName	The name of the associated queue on the service integration bus.		
DeliveryMode	The delivery mode for messages sent to this destination. This controls the persistence of messages on this destination.	Application, Persistent, NonPersistent	
TimeToLive	The default length of time in milliseconds from its dispatch time that a message sent to this destination should be kept by the system.		
Priority	The relative priority for messages sent to this destination, in the range 0 to 9, with 0 as the lowest priority and 9 as the highest priority.		
ReadAhead	Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.	AlwaysOn, AlwaysOff, AsConnection, Default	AsConnection
BusName	The name of the service integration bus to connect to.		
ScopeToLocalQP	Sets whether the service integration bus queue destination identified by this Queue is dynamically scoped to a single queue point if one exists on the messaging engine that the application is connected to.	On, Off	Off

Table 58. Queue properties (continued). The first column of the table lists the queue property names. The second column provides the description of the queue properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
ProducerPreferLocal	Sets whether a MessageProducer for this Queue should prefer a locally connected queue point of the service integration bus queue destination over any other queue points.	On, Off	On
ProducerBind	Set whether messages sent by a single MessageProducer to this Queue will go to the same service integration bus queue point, or whether no such restriction will be applied, and different messages will be sent to different queue points.	On, Off	Off
GatherMessages	Set whether messages on all service integration bus queue points or only a single queue point are visible to MessageConsumers and QueueBrowsers that use this Queue.	On, Off	Off

Table 59. Topic properties. The first column of the table lists the topic property names. The second column provides the description of the topic properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default
TopicSpace	The name of the topic space that contains the topic, on the service integration bus defined by the BusName property.		Default.Topic.Space
TopicName	The name of the topic that this JMS topic is assigned to, in the topic space defined by the TopicSpace property		
DeliveryMode	The delivery mode for messages sent to this destination. This controls the persistence of messages on this destination.	Application, Persistent, NonPersistent	
TimeToLive	The default length of time in milliseconds from its dispatch time that a message sent to this destination should be kept by the system.		
Priority	The relative priority for messages sent to this destination, in the range 0 to 9, with 0 as the lowest priority and 9 as the highest priority.		
ReadAhead	Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.	AlwaysOn, AlwaysOff, AsConnection, Default	AsConnection
BusName	The name of the service integration bus to connect to.		

Table 60. Activation configuration properties. The first column of the table lists the activation configuration property names. The second column provides the description of the activation configuration properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default	Required/optional
Destination	The name of the destination on the service integration bus.			Required
ProviderEndpoints	The list of comma separated endpoints used to connect to a bootstrap server.			Required
DestinationType	Whether the message-driven bean uses a queue or topic destination.	javax.jms.Queue, javax.jms.Topic		Required
BusName	The name of the service integration bus to connect to.			Required
MessageSelector	The JMS message selector used to determine which messages the message-driven bean receives. The value is a string that is used to select a subset of the available messages. The syntax is based on a subset of the SQL 92 conditional expression syntax, as described in the JMS specification.			Optional
AcknowledgeMode	How the session acknowledges any messages it receives.	Auto-acknowledge, Dups-ok-acknowledge	Auto-acknowledge	Optional
SubscriptionDurability	Whether a JMS topic subscription is durable or nondurable.	Durable, Nondurable	Nondurable	Optional
SubscriptionName	The subscription name needed for durable topic subscriptions. Required field when using a durable topic subscription.			Optional
MaxBatchSize	The maximum number of messages received from the messaging engine in a single batch.	1 through 2147483647	1	Optional
MaxConcurrency	The maximum number of endpoints to which messages are delivered concurrently	1 through 2147483647	10	Optional
RetryInterval	The delay (in seconds) between attempts to connect to a messaging engine.	1 through 2147483647	30	Optional
UserName				Optional

Table 60. Activation configuration properties (continued). The first column of the table lists the activation configuration property names. The second column provides the description of the activation configuration properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default	Required/optional
Password				Optional
DurableSubscriptionHome	The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS connection factory.			Optional
ShareDurableSubscriptions	Controls whether durable subscriptions are shared across connections with members of a server cluster.	InCluster, AlwaysShared, NeverShared	InCluster (always resolves to AlwaysOff as the client is always outside of a WebSphere Application Server clustered server)	Optional
ClientID	The JMS client identifier needed for durable topic subscriptions on all connections created by using this connection factory.			Optional
TargetTransportChain	The name of the protocol that resolves to a group of messaging engines.			Optional
ReadAhead	Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.	AlwaysOn, AlwaysOff, Default	Default	Optional
Target	The name of a target that identifies a group of messaging engines. Specify the type of target by using the Target type property.			Optional
TargetType	The type of target named in the Target property.	BusMember, Custom, ME	BusMember	Optional
TargetSignificance	This property specifies the significance of the target group.	Required, Preferred	Required	Optional
TopicSpace	The name of the topic space that contains the topic, on the service integration bus defined by the BusName property.		Default.Topic.Space	Optional

Table 60. Activation configuration properties (continued). The first column of the table lists the activation configuration property names. The second column provides the description of the activation configuration properties. The third column provides the permitted values for the properties if they are available. The fourth column includes the default values if they are available for the properties.

Property name	Description	Permitted values	Default	Required/optional
ForwarderDoesNotModify PayloadAfterSet	When enabled, Object/Bytes messages forwarded through this activation specification that have their payload modified will not have the data copied when it is set into the message and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.	true, false	false	Optional

Deploying inbound connections for the Resource Adapter for JMS with WebSphere Application Server

When deploying inbound connections, you must configure message-driven beans to use the Resource Adapter for JMS with WebSphere Application Server. A JMS activation configuration associated with one or more message-driven beans provides the configuration necessary for them to receive messages.

About this task

You can deploy message-driven beans within your application with a JMS activation configuration to access the Resource Adapter for JMS with WebSphere Application Server connection factories and destinations. When the message-driven bean is started, it uses the resource adapter to connect to the service integration bus, provided that the required messaging engine is available in WebSphere Application Server.

The Resource Adapter for JMS with WebSphere Application Server supports full two-phase XA transactional connections but it might also be run using local transactions or no transaction connections.

Procedure

1. Configure the following properties for the activation configuration:
   - Destination
   - ProviderEndpoints
   - DestinationType
   - BusName

   The Destination property value is the name of the destination from which the message-driven bean will be receiving messages.

   See “Configuration properties for the Resource Adapter for JMS with WebSphere Application Server” on page 607 for a description of these property names and other properties that might also be configured.
2. Configure the following additional properties if the message-driven bean will use XA resources over the Resource Adapter for JMS with WebSphere Application Server connection.
   - TargetType must be set to "ME"
   - TargetSignificance must be set to "Required"
   - Target value must be the name of the required ME

See “Configuration properties for the Resource Adapter for JMS with WebSphere Application Server” on page 607 for a description of these property names and other properties that might also be configured.

These properties permit the recovery of indoubt transactions, should this be necessary. For further information about indoubt transactions, see ../ae/tjm0165_.dita.

Default messaging provider [Settings]

A JMS provider enables messaging based on the Java Messaging Service (JMS). It provides Java EE connection factories to create connections for JMS destinations. This panel is used to manage the default messaging provider and its JMS resources.

To view this page in the console, click the following path:

Resources -> JMS -> JMS providers -> *a_messaging_provider*.

The default messaging provider uses service integration technologies to supply the messaging infrastructure. For example, the JMS queues and topics are assigned to destinations on a service integration bus.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Scope:

Specifies the highest topological level at which application servers can use this resource object.

Required	No
Data type	String

Name:

The name of the resource provider.

Required	No
Data type	String

Description:

A description of the resource adapter.

Required	No
Data type	Text area

JMS activation specification [Settings]

A JMS activation specification is associated with one or more message-driven beans and provides the configuration necessary for them to receive messages.

To view this page in the console, click one of the following paths:

- **Resources -> JMS -> Activation specifications -> *activation_specification_name***
- **Resources -> JMS -> JMS providers -> *a_messaging_provider* -> [Additional Properties] Activation specifications -> *activation_specification_name***

Use this panel to browse or change the configuration properties of the selected JMS activation specification for use with the default messaging provider.

You create a JMS activation specification if you want to use a message-driven bean to communicate with the default messaging provider through Java EE Connector Architecture (JCA) 1.5. JCA provides Java connectivity between application servers such as WebSphere Application Server, and enterprise information systems. It provides a standardized way of integrating JMS providers with Java EE application servers, and provides a framework for exchanging data with enterprise systems, where data is transferred in the form of messages.

All the activation specification configuration properties apart from **Name**, **JNDI name**, **Destination JNDI name**, and **Authentication alias** are overridden by appropriately named activation-configuration properties in the deployment descriptor of an associated EJB 2.1 or later message-driven bean. For an EJB 2.0 message-driven bean, the **Destination type**, **Subscription durability**, **Acknowledge mode** and **Message selector** properties are overridden by the corresponding elements in the deployment descriptor. For either type of bean the **Destination JNDI name** property can be overridden by a value specified in the message-driven bean bindings.

The activation specification properties influence how the default messaging provider chooses the messaging engine to which your message-driven bean application connects. By default, the environment automatically connects applications to an available messaging engine on the bus. However you can specify extra configuration details to influence the connection process; for example to identify special bootstrap servers, or to limit connection to a subgroup of available messaging engines, or to improve availability or performance, or to ensure sequential processing of messages received. For information about why and how to do this, see the topic How JMS applications connect to a messaging engine on a bus.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Scope:

Specifies the highest topological level at which application servers can use this resource object.

Required	No
Data type	String

Provider:

Specifies a JMS provider, which enables asynchronous messaging based on the Java Message Service (JMS). It provides J2EE connection factories to create connections for specific JMS queue or topic destinations. JMS provider administrative objects are used to manage JMS resources for the associated JMS provider.

Required No
Data type String

Name:

The required display name for the resource.

Required Yes
Data type String

JNDI name:

The JNDI name for the resource.

Type the JNDI name that is specified in the bindings for message-driven beans associated with this activation specification.

Required Yes
Data type String

Description:

An optional description for the resource.

Required No
Data type Text area

Destination type:

Whether the message-driven bean uses a queue or topic destination.

Required Yes
Data type drop-down list
Range

Queue The message-driven bean uses a JMS queue. The JNDI name of the JMS queue is specified on the **Destination JNDI name** property.

Topic The message-driven bean uses a JMS topic. The JNDI name of the JMS topic is specified on the **Destination JNDI name** property.

Destination JNDI name:

JNDI Name of destination

Type the JNDI name that the message-driven bean uses to look up the JMS destination in the JNDI namespace.

Select the type of destination on the “Destination type” on page 618 property.

Required	Yes
Data type	String

Message selector:

The JMS message selector used to determine which messages the message-driven bean receives. The value is a string that is used to select a subset of the available messages. The syntax is based on a subset of the SQL 92 conditional expression syntax, as described in the JMS specification. Refer to the information center for more information.

For example:

```
JMSType='car' AND color='blue' AND weight>2500
```

The selector string can refer to fields in the JMS message header and fields in the message properties. Message selectors cannot reference message body values.

A null value (an empty string) indicates that there is no message selector for the message consumer.

Required	No
Data type	String

Bus name:

The name of the bus to connect to.

Type the name of the service integration bus to which connections are made. This must be the name of the bus on which the destination identified by the “Destination JNDI name” on page 618 property is defined.

Required	No
Data type	Custom

Acknowledge mode:

How the session acknowledges any messages it receives.

The acknowledge mode indicates how a message received by a message-driven bean should be acknowledged.

Note:

The acknowledgement is sent when the message is deleted.

If you have a non-transactional message-driven bean, the system either deletes the message when the bean starts, or when the bean completes. If the bean generates an exception, and therefore does not complete, the system takes one of the following actions:

- If the system is configured to delete the message when the bean completes, then the message is despatched to a new instance of the bean, so the message has another opportunity to be processed.
- If the system is configured to delete the message when the bean starts, then the message is lost.

The message is deleted when the bean starts if the quality of service is set to Best effort nonpersistent. For all other qualities of service, the message is deleted when the bean completes.

Required No
Data type drop-down list
Range

Auto-acknowledge
The session automatically acknowledges the delivery of a message.

Duplicates-ok auto-acknowledge
The session lazily acknowledges the delivery of messages, which can improve performance, but can lead to a message-driven bean receiving a message more than once.

Target:

The name of a target that identifies a group of messaging engines. Specify the type of target using the Target type property.

Required No
Data type String

Target type:

The type of target named in the Target property.

Required No
Data type drop-down list
Range

Bus member name
The name of a bus member. This option retrieves the active messaging engines that are hosted by the named bus member (an application server or server cluster).

Custom messaging engine group name
The name of a custom group of messaging engines (that form a self-declaring cluster). This option retrieves the active messaging engines that have registered with the named custom group.

Messaging engine name
The name of a messaging engine. This option retrieves the available endpoints that can be used to reach the named messaging engine.

Target significance:

This property specifies the significance of the target group.

Required No
Data type drop-down list

Range

Preferred
It is preferred that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, a messaging engine outside the target group is selected if available in the same service integration bus.

Required
It is required that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, the connection process fails.

Target inbound transport chain:

The name of the inbound transport chain that the application should target when connecting to a messaging engine in a separate process to the application. If a messaging engine in another process is chosen, a connection can be made only if the messaging engine is in a server that runs the specified inbound transport chain. Refer to the information center for more information.

If the selected messaging engine is in the same server as the application, a direct in-process connection is made and this transport chain property is ignored.

The transport chains represent network protocol stacks operating within a server. The name you specify must be one of the transport chains available in the server that hosts the messaging engine, as listed on the **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports** panel. The following transport chains are provided, but you can define your own transport chains on that panel.

InboundBasicMessaging
This is a connection-oriented protocol that uses a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging
This is the InboundBasicMessaging protocol wrapped in SSL.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required No
Data type String

Provider endpoints:

A comma-separated list of endpoint triplets, with the syntax hostName:portNumber:chainName, used to connect to a bootstrap server. For example Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging.

Provider endpoints are not used unless the specified bus cannot be found in the local cell. Message-driven bean (MDB) applications first attempt to connect the specified bus in the local cell. If this attempt fails, provider endpoints are used to allow the applications to consume messages from a remote cell.

If the host name is not specified, `localhost` is used as a default value.

If the port number is not specified, `7276` is used as the default value.

If the protocol is not specified, a predefined chain such as `BootstrapBasicMessaging` is used as the default value.

Required	No
Data type	Text area

Maximum batch size:

The maximum number of messages received from the messaging engine in a single batch.

The maximum number of messages in a single batch delivered serially to a single message-driven bean instance. Batching of messages can improve performance particularly when used with Acknowledge mode set to Duplicates-ok auto-acknowledge. If message-ordering must be retained across failed deliveries, set the batch size to `1`.

Required	No
Data type	Integer
Range	1 through 2147483647

Maximum concurrent MDB invocations per endpoint:

The maximum number of endpoints to which messages are delivered concurrently.

Increasing this number can improve performance but can increase the number of threads that are in use at any one time. If message ordering must be retained across failed deliveries, set the maximum concurrent endpoints to `1`. Message ordering applies only if the destination that the message-driven bean is consuming from is not a partitioned destination. Partitioned destinations are used in a workload sharing scenario in a cluster.

Required	No
Data type	Integer
Range	1 through 2147483647

Automatically stop endpoints on repeated message failure:

These parameters enable an endpoint to stop automatically when the number of sequentially failing messages reaches a limit that you specify. This helps to distinguish between one or two messages that fail because of problems with the messages themselves, and a system resource problem that results in many sequentially failing messages.

Stopping the endpoint reduces the number of messages being moved unnecessarily to an exception destination when the problem is not caused by messages that are failing to be processed.

When an endpoint is stopped automatically, its Status on the administrative console panel is red. It must be restarted manually by clicking **Resume**.

After an endpoint is restarted, any failing messages that caused the endpoint to be stopped are retried. If they continue to fail they are moved to an exception destination, if configured.

Enable:

Enable automatic stopping of an endpoint based on the parameters below.

Required	No
Data type	Boolean

Sequential failed message threshold:

The endpoint will be stopped when the number of sequentially failing messages reaches the configured limit. Due to processing dependencies in the MDB the actual number of messages processed may exceed this value.

This property is not enabled unless the Automatically stop endpoints on repeated message failure property is enabled.

Required	No
Data type	Integer

Delay between failing message retries:

Any message that fails to be processed by the MDB but has not reached its maximum failed delivery limit will only be retried after this period of time (in milliseconds) has passed. Other messages may be tried during this period, unless the sequential failure threshold and the maximum concurrency is set to 1.

Setting a delay between failing message retries reduces the number of messages unnecessarily moved to the exception destination before the MDB is stopped. To minimize the number of messages that are moved, make this delay greater than the expected time interval between messages arriving on the destination.

This property is not enabled unless the Automatically stop endpoints on repeated message failure property is enabled.

Required	No
Data type	Integer
Range	The time in milliseconds. A value of `0` indicates no delay between retries.

Subscription durability:

Whether a JMS topic subscription is durable or nondurable

Usually, only one application at a time can have a consumer for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers.

Required	No
Data type	drop-down list

Range

Durable
The messaging provider stores messages while the message-driven bean is not available, and delivers the messages when the message-driven bean becomes available again.

Nondurable
The messaging provider does not store and redeliver messages if a message-driven bean is not available.

Subscription name:

The subscription name needed for durable topic subscriptions. Required field when using a durable topic subscription.

Each JMS durable subscription is identified by a subscription name (specified on this property). A JMS connection also has an associated client identifier (specified on the Client identifier property), which is used to associate a connection and its objects with the list of messages (on the durable subscription) that is maintained by the JMS provider for the client.

This subscription name must be unique within a given client identifier.

Required No
Data type String

Client identifier:

The JMS client identifier needed for durable topic subscriptions on all connections created using this activation specification.

The value specified is a unique identifier for a client (message-driven bean). The client identifier is used to associate a client connection with the list of messages (on a durable subscription) that the messaging provider keeps for the client. When a client becomes available again, after a being unavailable, the messaging provider uses the client identifier to redeliver stored messages to the correct client.

Required No
Data type String

Durable subscription home:

The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS activation specification. This is a required field when using a durable topic subscription.

Administrators can manage the runtime state of durable subscriptions through publication points for this messaging engine.

Required No
Data type String

Pass message payload by reference:

When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant.If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION:
The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

Applications that use this activation specification to receive messages must obey the following rule::

- The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When enabled, Object Messages received through this activation spec will only have their message data serialized by the system when absolutely necessary. The data obtained from those messages must be treated as readOnly by applications.

Required	No
Data type	Boolean

Applications resending messages that were originally received using this activation specification must obey the following rules::

- The application can replace the data object in a JMS object message, provided that the data object has not yet been set in the message. The application does not modify or replace the data object after it is set in the message.
- The application can replace the byte array in a JMS bytes message, but only by using a single call to writeBytes(byte[]), and provided that the byte array has not yet been set in the message. The application does not modify or replace the byte array after it is set in the message.

When enabled, Object/Bytes Messages forwarded through this activation specification that have their payload modified will not have the data copied when it is set into the message and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.

Required	No
Data type	Boolean

Share durable subscriptions:

Controls whether or not durable subscriptions are shared across connections with members of a server cluster.

Usually, only one session at a time can have a TopicSubscriber for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers, one on each application server in the server cluster.

This option should be changed from its default only in WebSphere Application Server environments that support server clusters.

Required	No

Data type drop-down list

Range

In cluster
Allows sharing of durable subscriptions when connections are made from within a server cluster.

Always shared
Durable subscriptions can be shared across connections.

Never shared
Durable subscriptions are never shared across connections.

Share data source with CMP:

Allow sharing of connections between JMS and container-managed persistence (CMP) entity beans.

This option is used as part of the task to enable container-managed persistence (CMP) entity beans to share the database connections used by the data store of a messaging engine. This has been estimated as a potential performance improvement of 15% for overall message throughput, but can only be used for entity beans connected to the application server that contains the messaging engine.

For more information about using this option, see the topic Enabling CMP entity beans and messaging engine data stores to share database connections.

Required No
Data type Boolean

Read ahead:

Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

You can override this property for individual JMS destinations by setting the **Read ahead** property on the JMS destination.

Required No
Data type drop-down list

Range

Default The message provider preemptively assigns messages to consumers on nondurable subscriptions and unshared durable subscriptions. That is, read ahead optimization is turned on only when there can only be a single consumer.

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Disabled
The messaging provider does not preemptively assign messages to consumers.

Always activate MDBs in all servers:

This property is only used when the MDB application is running on a server that is a member of the bus that the application is targeting. It has no effect when the MDB is running on a server that is not a member of the target bus.

If the MDB application is running on a server that is a member of the target bus, enabling this option allows the MDB application to process messages whether or not the server also hosts a running messaging engine. If this option is not enabled, then MDB applications on servers that do not have a local ME running do not process messages.

Required No
Data type Boolean

Retry interval:

The delay (in seconds) between attempts to connect to a messaging engine, both for the initial connection, and any subsequent attempts to establish a better connection.

Required No
Data type Integer
Range 1 through 2147483647

Authentication alias

The name of a J2C authentication alias used for component-managed authentication of connections to the service integration bus.

A Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) authentication alias specifies the user ID and password that is used to authenticate the creation of a new connection to the JMS provider.

Required No
Data type drop-down list

Related Items

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

Buses
A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Default messaging provider unified connection factory [Settings]

A JMS connection factory is used to create connections to the associated JMS provider of JMS destinations, for both point-to-point and publish/subscribe messaging. Use connection factory administrative objects to manage JMS connection factories for the default messaging provider.

To view this page in the console, click one of the following paths:

- **Resources -> JMS -> Connection factories -> *factory_name***
- **Resources -> JMS -> JMS providers -> *a_messaging_provider* -> [Additional Properties] Connection factories -> *factory_name***

Set, browse or change the configuration properties of a JMS connection factory for use with the default messaging JMS provider. These configuration properties control how connections are created to associated JMS queues and topics.

By default, connections created by using this JMS connection factory in the server containers (for example, from an enterprise bean) are pooled by using Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection pooling. You can modify the connection pool settings for this connection factory by selecting **Connection pool properties** in the Additional properties section of the administrative console panel.

The connection factory properties influence how the default messaging provider chooses the messaging engine to which your JMS application connects. By default, the environment automatically connects applications to an available messaging engine on the bus. However you can specify extra configuration details to influence the connection process; for example to identify special bootstrap servers, or to limit connection to a subgroup of available messaging engines, or to improve availability or performance, or to ensure sequential processing of messages received. For information about how to do this, see the topic Configuring the messaging engine selection process for JMS applications.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Scope:

Specifies the highest topological level at which application servers can use this resource object.

Required	No
Data type	String

Provider:

Specifies a JMS provider, which enables asynchronous messaging based on the Java Message Service (JMS). It provides J2EE connection factories to create connections for specific JMS queue or topic destinations. JMS provider administrative objects are used to manage JMS resources for the associated JMS provider.

Required	No
Data type	String

Name:

The required display name for the resource.

Required	Yes
Data type	String

JNDI name:

The JNDI name for the resource.

Required	Yes
Data type	String

Description:

An optional description for the resource.

Required	No
Data type	Text area

Category:

An optional category string to use when classifying or grouping the resource.

Required	No
Data type	String

Bus name:

The name of the service integration bus to connect to.

Enter the name of the local bus in situations where an application makes connection to foreign buses.

Required	Yes
Data type	Custom

Target:

The name of a target that identifies a group of messaging engines. Specify the type of target using the Target type property.

Before the connection proximity search is performed to select a suitable messaging engine, the set of messaging engines that are members of the specified target group are selected. The connection proximity

search is then restricted to these messaging engines. If a target group is not specified (the default), then all messaging engines in the bus are considered during the connection proximity search.

For example, if the Target type property is set to Bus member name, the Target property specifies the name of the bus member from which suitable messaging engines can be chosen.

Required No
Data type String

Target type:

The type of target named in the Target property.

Required No
Data type drop-down list
Range

Bus member name
The name of a bus member. This option retrieves the active messaging engines that are hosted by the named bus member (an application server or server cluster).

To specify a non-clustered bus member the Target property must be set to `<Node01>.<server1>`, for example Node01.server1. For a cluster bus member the Target property must be set to the cluster name.

Custom messaging engine group name
The name of a custom group of messaging engines (that form a self-declaring cluster). This option retrieves the active messaging engines that have registered with the named custom group.

Messaging engine name
The name of a messaging engine. This option retrieves the available endpoints that can be used to reach the named messaging engine.

Target significance:

This property specifies the significance of the target group.

This property defines whether the connection proximity search is restricted to only the messaging engines in the target group.

Required No
Data type drop-down list

Range

Preferred

It is preferred that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, a messaging engine outside the target group is selected if available in the same service integration bus.

Note: A connection to a non-preferred target might be returned even if a preferred one is available. This can happen when connection pooling is enabled for a ConnectionFactory, which it is by default when you use a JMS ConnectionFactory in a server environment:

- When a preferred messaging engine is not available, a connection to a non-preferred one can be created and stored in the connection pool.
- The next time the application requests a connection it receives this pooled connection even if the preferred messaging engine has subsequently become available.

You can modify the connection pool settings to regularly discard all unused connections in the pool. After the connection pool is emptied, connections are made to the preferred messaging engine if one is available. For example, set the **ReapTime**, **AgedTimeout** and **UnusedTimeout** to 300 seconds, and the **PurgePolicy** to EntirePool. This refreshes the connection pool every 5 minutes, after which time the application selects a preferred messaging engine if one is available.

Required

It is required that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, the connection process fails.

Target inbound transport chain:

The name of the inbound transport chain that the application should target when connecting to a messaging engine in a separate process to the application. If a messaging engine in another process is chosen, a connection can be made only if the messaging engine is in a server that runs the specified inbound transport chain. Refer to the information center for more information.

These transport chains specify the communication protocols that can be used to communicate with the application server to which the client application is connected.

If the selected messaging engine is in the same server as the application, a direct in-process connection is made and this transport chain property is ignored.

The transport chains represent network protocol stacks operating within a server. The name you specify must be one of the transport chains available in the server that hosts the messaging engine, as listed on the **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports** panel. The following transport chains are provided, but you can define your own transport chains on that panel.

InboundBasicMessaging
This is a connection-oriented protocol that uses a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging
This is the InboundBasicMessaging protocol wrapped in SSL.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required	No
Data type	String

Provider endpoints:

A comma-separated list of endpoint triplets, with the syntax hostName:portNumber:chainName, used to connect to a bootstrap server. For example Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging. If hostName is not specified, the default is localhost. If portNumber is not specified, the default is 7276. If chainName is not specified, the default is BootstrapBasicMessaging. Refer to the information center for more information.

You only have to modify this property if you have client applications running outside of an application server, or applications on a server in another cell, that want to use this connection factory to connect to the target service integration bus specified on the connection factory.

To use JMS destinations of the default messaging provider, an application connects to a messaging engine on the target service integration bus to which the destinations are assigned. For example, a JMS queue is assigned to a queue destination on a service integration bus.

Client applications running outside of an application server - for example, running in a client container or outside the WebSphere Application Server environment - cannot locate directly a suitable messaging engine to connect to in the target bus. Similarly, an application running on a server in one cell to connect to a target bus in another cell cannot locate directly a suitable messaging engine to connect to in the target bus.

In these scenarios, the clients (or servers in another bus) must complete a bootstrap process through a *bootstrap server* that is a member of the target bus. A bootstrap server is an application server running the SIB Service, but does not have to be running any messaging engines. The bootstrap server selects a messaging engine that is running in an application server that supports the required *target transport chain*. For the bootstrap process to be possible, you must configure one or more *provider end points* in the connection factory used by the client.

A bootstrap server uses a specific port and bootstrap transport chain. The port is the **SIB_ENDPOINT_ADDRESS** (or **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled), of the messaging engine that hosts the remote end of the link. Together with host name, these form the *endpoint address* of the bootstrap server.

The properties of a JMS connection factory used by an application control the selection of a suitable messaging engine and how the application connects to the selected messaging engine.

- If no security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7276 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used
- If security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7286 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used

Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

If you want an application to use a bootstrap server with a different endpoint address, you must specify the required endpoint address on the **Provider endpoints** property of the JMS connection factories that the client application uses. You can specify one or more endpoint addresses of bootstrap servers.

The endpoint addresses for bootstrap servers must be specified in every JMS connection factory that is used by applications outside of an application server. To avoid having to specify a long list of bootstrap servers, you can provide a few highly-available servers as dedicated bootstrap servers. Then you only have to specify a short list of bootstrap servers on each connection factory.

Note: When configuring a connection to a non-default bootstrap server, specify the required values for the endpoint address and use colons as separators.

For example: for a server assigned non-secure port 7278, on host boothost1, that uses the default transport chain BootstrapBasicMessaging:

```
boothost1:7278:BootstrapBasicMessaging
or
boothost1:7278
```

and for a server assigned secure port 7289, on host boothost2, that uses the predefined transport chain BootstrapTunneledSecureMessaging:

```
boothost2:7289:BootstrapTunneledSecureMessaging
```

The syntax for an endpoint address is as follows:

```
[ [host_name] [ ":" [port_number] [ ":" chain_name] ] ]
```

where:

host_name
: is the name of the host on which the server runs. It can be an IP address. For an IPv6 address, put square braces ([]) around *host_name* as shown in the example below:

  ```
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
  ```

  . If a value is not specified, the default is localhost.

Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

port_number
where specified, is one of the following addresses of the messaging engine that hosts the remote end of the link:

- **SIB_ENDPOINT_ADDRESS** if security is not enabled
- For secure connections, **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled.

If *port_number* is not specified, the default is 7276.

To find either of these values by using the administrative console, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Communications] Ports**.

chain_name
is the name of a predefined bootstrap transport chain used to connect to the bootstrap server. If not specified, the default is BootstrapBasicMessaging.

The following predefined bootstrap transport chains are provided:

BootstrapBasicMessaging
This corresponds to the server transport chain InboundBasicMessaging (JFAP-TCP/IP)

BootstrapSecureMessaging
This corresponds to the server transport chain InboundSecureMessaging (JFAP-SSL-TCP/IP)

BootstrapTunneledMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.) This transport chain tunnels JFAP and uses HTTP wrappers.

BootstrapTunneledSecureMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See**Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.) This transport chain tunnels JFAP and uses HTTP wrappers.

Specifying *host_name* : *chain_name* instead of *host_name* : : *chain_name* (with two colons) is incorrect. It is valid to enter nothing, or to enter any of the following: "a", "a:", ":7276", "::chain", and so on. The default value applies if you do not specify a value, but you must separate the fields with ":"s.

If you want to provide more than one bootstrap server, identify all the required endpoint addresses. Separate each endpoint address by a comma character. For example, to use the servers from the earlier example:

```
boothost1:7278:BootstrapBasicMessaging,
  boothost2:7289:BootstrapTunneledSecureMessaging,
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
```

Required	No
Data type	Text area

Connection proximity:

The proximity of messaging engines that can accept connection requests, in relation to the bootstrap messaging engine.

When a client issues a client connect request, the processing attaches to the required bus according to the following logic:

- If a target group is specified, connect to the first messaging engine that satisfies the following conditions for the target type:
  - **Server** Look for a messaging engine in the same server.
  - **Cluster** Look for a messaging engine in the same server, then on other servers in the same cluster.
  - **Host** Look for a messaging engine in the same server, then on other servers in the same cluster, then on other servers in the same host.
  - **Bus** Look for a messaging engine in the same server, then on other servers in the same cluster, then on other servers in same host, then any other messaging engine on the same bus.
- If a target group is not specified, or a target group is specified but no suitable messaging engine is found and target significance is Preferred, connect to the first messaging engine that satisfies the following conditions for the target type:
  - **Server** Look for a messaging engine in the same server.
  - **Cluster** Connection fails.
  - **Host** Look for a messaging engine in the same server, then on other servers in the same host.
  - **Bus** Look for a messaging engine in the target group in same server, then on other servers in same host, then any other messaging engine on the same bus.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required No
Data type drop-down list
Range

Bus Connections can be made to messaging engines in the same bus.

Cluster Connections can be made to messaging engines in the same server cluster.

Host Connections can be made to messaging engines in the same host.

Server Connections can be made to messaging engines in the same application server.

Client identifier:

The JMS client identifier needed for durable topic subscriptions on all connections created using this connection factory. This identifier is required if the application is doing durable pub/sub

Required No
Data type String

Durable subscription home:

The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS connection factory.

Required No
Data type Custom

Nonpersistent message reliability:

The reliability applied to nonpersistent JMS messages sent using this connection factory.

You can change the delivery reliability option for the destination of a message that is sent by a JMS application as Nonpersistent. The default is Express nonpersistent but you have a range of other options, including those with persistent characteristics, with Assured persistent being the most reliable. For more information see the topic Message reliability levels - JMS delivery mode and service integration quality of service.

Required No

Data type drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

As bus destination
Use the delivery option configured for the bus destination.

Persistent message reliability:

The reliability applied to persistent JMS messages sent using this connection factory.

You can change the delivery reliability option for the destination of a message that is sent by a JMS application as Persistent. The default is Reliable persistent but you have a range of other options including those with nonpersistent characteristics, with Best effort nonpersistent being the least reliable. For more information see the topic Message reliability levels - JMS delivery mode and service integration quality of service.

Important: If you change the delivery reliability options of a message sent by a JMS application from one of the **Persistent message reliability** options (Assured persistent and Reliable persistent) to one of the **Nonpersistent message reliability** options (Best effort nonpersistent, Express nonpersistent, and Reliable nonpersistent), you risk losing messages in certain circumstances. For example, at server restart, or when there is heavy workload.

Required No

Data type drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

As bus destination
Use the delivery option configured for the bus destination.

Read ahead:

Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

You can override this property for individual JMS destinations by setting the **Read ahead** property on the JMS destination.

Required No
Data type drop-down list

Range

Default The message provider preemptively assigns messages to consumers on nondurable subscriptions and unshared durable subscriptions. That is, read ahead optimization is turned on only when there can only be a single consumer.

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Disabled
The messaging provider does not preemptively assign messages to consumers.

Temporary queue name prefix:

The prefix of up to twelve characters used for names of temporary queues created by applications that use this connection factory.

Required No
Data type String

Temporary topic name prefix:

The prefix of up to twelve characters used for names of temporary topics created by applications that use this connection factory.

Required No
Data type String

Share durable subscriptions:

Controls whether or not durable subscriptions are shared across connections with members of a server cluster.

Usually, only one session at a time can have a TopicSubscriber for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers.

Required No
Data type drop-down list

Range

In cluster
Allows sharing of durable subscriptions when connections are made from within a server cluster.

Always shared
Durable subscriptions can be shared across connections.

Never shared
Durable subscriptions are never shared across connections.

Pass message payload by reference: When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION:
The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

Applications that use this connection factory to send messages must obey the following rules::

- The application does not modify the data object contained in a JMS object message.
- The application populates a JMS bytes message by using a single call to writeBytes(byte[]) and does not modify the byte array after it is set in the message.

When enabled, Object/Bytes Messages sent by a message producing application that has connected to the bus using this connection factory will not have their data copied when set and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.

Required No
Data type Boolean

Applications that use this connection factory to receive messages must obey the following rule::

- The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When enabled, Object Messages received by a message consuming application that has connected to this connection factory will only have their message data serialized by the system when absolutely necessary. The data obtained from these messages must be treated as readOnly by applications.

Required No
Data type Boolean

Log missing transaction contexts:

Whether or not the container logs that there is a missing transaction context when a connection is obtained.

The Java EE programming model indicates that connections should always have a transaction context. However, some applications do not correctly have a transaction context associated with them.

Select this property to log connections being created without a transaction context.

Required	No
Data type	Boolean

Manage cached handles:

Whether cached handles (handles held in instance variables in a bean) should be tracked by the container.

Select this option to track handle management, which can be useful for debugging purposes. However, tracking handles can significantly reduce performance if used at run time.

Required	No
Data type	Boolean

Share data source with CMP:

Allow sharing of connections between JMS and container-managed persistence (CMP) entity beans.

This option is used as part of the task to enable container-managed persistence (CMP) entity beans to share the database connections used by the data store of a messaging engine. This has been estimated as a potential performance improvement of 15% for overall message throughput, but can only be used for entity beans connected to the application server that contains the messaging engine.

For more information about using this option, see the topic Enabling CMP entity beans and messaging engine data stores to share database connections.

Required	No
Data type	Boolean

Authentication alias for XA recovery:

Specifies the alias that the connection factory uses to authenticate with the EIS for transaction recovery.

Select the alias to be used during transaction recovery processing.

This property provides a list of the JCA authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate during XA recovery processing.

If you have enabled security for the associated service integration bus, select the alias that specifies the user ID and password used for XA recovery that is valid in the user registry for WebSphere Application Server. This property must be set if bus security is enabled and XA transactions are to be used.

Required	No
Data type	drop-down list

Mapping-configuration alias:

Specifies the mapping configuration alias for the Java Authentication and Authorization Service (JAAS) mapping configuration that is used by this connection factory.

This field will be used only in the absence of a loginConfiguration on the component resource reference. The specification of a login configuration and associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is Container.If the DefaultPrincipalMapping login configuration is specified, the associated property will be a JAAS - J2C authentication data entry alias. See related item JAAS - J2C authentication data entry to define a new alias.

Required	No
Data type	drop-down list

Container-managed authentication alias:

This alias specifies a user ID and password to be used to authenticate connections to the JMS provider for container-managed authentication. This setting is only used when the res-auth value is container, and the authentication alias was not set when the application was deployed.

Required	No
Data type	drop-down list

Additional Properties

Connection pool properties
An optional set of connection pool settings.

Related Items

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

Buses
A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Default messaging provider queue connection factory [Settings]

A JMS queue connection factory is used to create connections to the associated JMS provider of JMS queues, for point-to-point messaging. Use queue connection factory administrative objects to manage JMS queue connection factories for the default messaging provider.

To view this page in the console, click one of the following paths:

- **Resources -> JMS -> Queue connection factories -> *factory_name***
- **Resources -> JMS -> JMS providers -> *a_messaging_provider* -> [Additional Properties] Queue connection factories -> *factory_name***

Use this panel to browse or change the configuration properties of the selected JMS queue connection factory for use with the default messaging JMS provider. These configuration properties control how connections are created to associated JMS queues.

By default, connections created by using this JMS connection factory in the server containers (for example, from an enterprise bean) are pooled by using Java Platform, Enterprise Edition (Java EE)

Connector Architecture (JCA) connection pooling. You can modify the connection pool settings for this connection factory by selecting **Connection pool properties** link in the Additional properties section of the administrative console panel.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Scope:

Specifies the highest topological level at which application servers can use this resource object.

Required	No
Data type	String

Provider:

Specifies a JMS provider, which enables asynchronous messaging based on the Java Message Service (JMS). It provides J2EE connection factories to create connections for specific JMS queue or topic destinations. JMS provider administrative objects are used to manage JMS resources for the associated JMS provider.

Required	No
Data type	String

Name:

The required display name for the resource.

Required	Yes
Data type	String

JNDI name:

The JNDI name for the resource.

As a convention, use a JNDI name of the form jms/*Name*, where *Name* is the logical name of the resource. For more information about the use of JNDI and its syntax, see the topic JNDI support in WebSphere Application Server.

Required	Yes
Data type	String

Description:

An optional description for the resource.

Required	No
Data type	Text area

Category:

An optional category string to use when classifying or grouping the resource.

Required	No
Data type	String

Bus name:

The name of the service integration bus to connect to.

This is the name of the service integration bus that this connection factory is used to create connections to.

Enter the name of the local bus in situations where an application makes connection to foreign buses.

Required	Yes
Data type	Custom

Target:

The name of a target that identifies a group of messaging engines. Specify the type of target using the Target type property.

Required	No
Data type	String

Target type:

The type of target named in the Target property.

This indicates the name of a target that is to be used to determine one or messaging engines to handle work. The type of target is indicated by the Target type property

Connections are load balanced across the available messaging engines that satisfy the selection criteria.

If want applications to be able to connect to any messaging engine in the bus, do not set this property.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required	No
Data type	drop-down list

Range

Bus member name
The name of a bus member. This option retrieves the active messaging engines that are hosted by the named bus member (an application server or server cluster).

To specify a non-clustered bus member the Target property must be set to `<Node01>.<server1>`, for example Node01.server1. For a cluster bus member the Target property must be set to the cluster name.

Custom messaging engine group name
The name of a custom group of messaging engines (that form a self-declaring cluster). This option retrieves the active messaging engines that have registered with the named custom group.

Messaging engine name
The name of a messaging engine. This option retrieves the available endpoints that can be used to reach the named messaging engine.

Target significance:

This property specifies the significance of the target group.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required No
Data type drop-down list
Range

Preferred
It is preferred that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, a messaging engine outside the target group is selected if available in the same service integration bus.

Required
It is required that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, the connection process fails.

Target inbound transport chain:

The name of the inbound transport chain that the application should target when connecting to a messaging engine in a separate process to the application. If a messaging engine in another process is chosen, a connection can be made only if the messaging engine is in a server that runs the specified inbound transport chain. Refer to the information center for more information.

If the selected messaging engine is in the same server as the application, a direct in-process connection is made and this transport chain property is ignored.

The transport chains represent network protocol stacks operating within a server. The name you specify must be one of the transport chains available in the server that hosts the messaging engine, as listed on the **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports** panel. The following transport chains are provided, but you can define your own transport chains on that panel.

InboundBasicMessaging
This is a connection-oriented protocol that uses a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging
This is the InboundBasicMessaging protocol wrapped in SSL.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required	No
Data type	String

Provider endpoints:

A comma-separated list of endpoint triplets, with the syntax hostName:portNumber:chainName, used to connect to a bootstrap server. For example Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging. If hostName is not specified, the default is localhost. If portNumber is not specified, the default is 7276. If chainName is not specified, the default is BootstrapBasicMessaging. Refer to the information center for more information.

You only have to modify this property if you have client applications running outside of an application server, or applications on a server in another cell, that want to use this connection factory to connect to the target service integration bus specified on the connection factory.

To use JMS destinations of the default messaging provider, an application connects to a messaging engine on the target service integration bus to which the destinations are assigned. For example, a JMS queue is assigned to a queue destination on a service integration bus.

Client applications running outside of an application server - for example, running in a client container or outside the WebSphere Application Server environment - cannot locate directly a suitable messaging engine to connect to in the target bus. Similarly, an application running on a server in one cell to connect to a target bus in another cell cannot locate directly a suitable messaging engine to connect to in the target bus.

In these scenarios, the clients (or servers in another bus) must complete a bootstrap process through a *bootstrap server* that is a member of the target bus. A bootstrap server is an application server running the SIB Service, but does not have to be running any messaging engines. The bootstrap server selects a messaging engine that is running in an application server that supports the required *target transport chain*. For the bootstrap process to be possible, you must configure one or more *provider end points* in the connection factory used by the client.

A bootstrap server uses a specific port and bootstrap transport chain. The port is the **SIB_ENDPOINT_ADDRESS** (or **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled), of the messaging engine that hosts the remote end of the link. Together with host name, these form the *endpoint address* of the bootstrap server.

The properties of a JMS connection factory used by an application control the selection of a suitable messaging engine and how the application connects to the selected messaging engine.

- If no security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7276 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used
- If security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7286 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used

Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

If you want an application to use a bootstrap server with a different endpoint address, you must specify the required endpoint address on the **Provider endpoints** property of the JMS connection factories that the client application uses. You can specify one or more endpoint addresses of bootstrap servers.

The endpoint addresses for bootstrap servers must be specified in every JMS connection factory that is used by applications outside of an application server. To avoid having to specify a long list of bootstrap servers, you can provide a few highly-available servers as dedicated bootstrap servers. Then you only have to specify a short list of bootstrap servers on each connection factory.

Note: When configuring a connection to a non-default bootstrap server, specify the required values for the endpoint address and use colons as separators.

For example: for a server assigned non-secure port 7278, on host boothost1, that uses the default transport chain BootstrapBasicMessaging:

```
boothost1:7278:BootstrapBasicMessaging
or
boothost1:7278
```

and for a server assigned secure port 7289, on host boothost2, that uses the predefined transport chain BootstrapTunneledSecureMessaging:

```
boothost2:7289:BootstrapTunneledSecureMessaging
```

The syntax for an endpoint address is as follows:

```
[ [host_name] [ ":" [port_number] [ ":" chain_name] ] ]
```

where:

host_name
: is the name of the host on which the server runs. It can be an IP address. For an IPv6 address, put square braces ([]) around *host_name* as shown in the example below:

  ```
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
  ```

  . If a value is not specified, the default is localhost.

Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

port_number
where specified, is one of the following addresses of the messaging engine that hosts the remote end of the link:

- **SIB_ENDPOINT_ADDRESS** if security is not enabled
- For secure connections, **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled.

If *port_number* is not specified, the default is 7276.

To find either of these values by using the administrative console, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Communications] Ports**.

chain_name
is the name of a predefined bootstrap transport chain used to connect to the bootstrap server. If not specified, the default is BootstrapBasicMessaging.

The following predefined bootstrap transport chains are provided:

BootstrapBasicMessaging
This corresponds to the server transport chain InboundBasicMessaging (JFAP-TCP/IP)

BootstrapSecureMessaging
This corresponds to the server transport chain InboundSecureMessaging (JFAP-SSL-TCP/IP)

BootstrapTunneledMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.) This transport chain tunnels JFAP and uses HTTP wrappers.

BootstrapTunneledSecureMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See**Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.) This transport chain tunnels JFAP and uses HTTP wrappers.

Specifying *host_name* : *chain_name* instead of *host_name* : : *chain_name* (with two colons) is incorrect. It is valid to enter nothing, or to enter any of the following: "a", "a:", ":7276", "::chain", and so on. The default value applies if you do not specify a value, but you must separate the fields with ":"s.

If you want to provide more than one bootstrap server, identify all the required endpoint addresses. Separate each endpoint address by a comma character. For example, to use the servers from the earlier example:

```
boothost1:7278:BootstrapBasicMessaging,
  boothost2:7289:BootstrapTunneledSecureMessaging,
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
```

Required	No
Data type	Text area

Connection proximity:

The proximity of messaging engines that can accept connection requests, in relation to the bootstrap messaging engine.

When a client issues a client connect request, the processing attaches to the required bus according to the following logic:

- If a target group is specified, connect to the first messaging engine that satisfies the following conditions for the target type:
  - **Server** Look for a messaging engine in the same server.
  - **Cluster** Look for a messaging engine in the same server, then on other servers in the same cluster.
  - **Host** Look for a messaging engine in the same server, then on other servers in the same cluster, then on other servers in the same host.
  - **Bus** Look for a messaging engine in the same server, then on other servers in the same cluster, then on other servers in same host, then any other messaging engine on the same bus.
- If a target group is not specified, or a target group is specified but no suitable messaging engine is found and target significance is Preferred, connect to the first messaging engine that satisfies the following conditions for the target type:
  - **Server** Look for a messaging engine in the same server.
  - **Cluster** Connection fails.
  - **Host** Look for a messaging engine in the same server, then on other servers in the same host.
  - **Bus** Look for a messaging engine in the target group in same server, then on other servers in same host, then any other messaging engine on the same bus.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required No
Data type drop-down list
Range

Bus Connections can be made to messaging engines in the same bus.

Cluster Connections can be made to messaging engines in the same server cluster.

Host Connections can be made to messaging engines in the same host.

Server Connections can be made to messaging engines in the same application server.

Nonpersistent message reliability:

The reliability applied to nonpersistent JMS messages sent using this connection factory.

You can change the delivery reliability option for the destination of a message that is sent by a JMS application as Nonpersistent. The default is Express nonpersistent but you have a range of other options, including those with persistent characteristics, with Assured persistent being the most reliable. For more information see the topic Message reliability levels - JMS delivery mode and service integration quality of service.

Required No
Data type drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

As bus destination
Use the delivery option configured for the bus destination.

Persistent message reliability:

The reliability applied to persistent JMS messages sent using this connection factory.

You can change the delivery reliability option for the destination of a message that is sent by a JMS application as Persistent. The default is Reliable persistent but you have a range of other options including those with nonpersistent characteristics, with Best effort nonpersistent being the least reliable. For more information see the topic Message reliability levels - JMS delivery mode and service integration quality of service.

Important: If you change the delivery reliability options of a message sent by a JMS application from one of the **Persistent message reliability** options (Assured persistent and Reliable persistent) to one of the **Nonpersistent message reliability** options (Best effort nonpersistent, Express nonpersistent, and Reliable nonpersistent), you risk losing messages in certain circumstances. For example, at server restart, or when there is heavy workload.

Required No
Data type drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

As bus destination
Use the delivery option configured for the bus destination.

Read ahead:

Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

You can override this property for individual JMS destinations by setting the **Read ahead** property on the JMS destination.

Required No
Data type drop-down list

Range

Default The message provider preemptively assigns messages to consumers on nondurable subscriptions and unshared durable subscriptions. That is, read ahead optimization is turned on only when there can only be a single consumer.

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Disabled
The messaging provider does not preemptively assign messages to consumers.

Temporary queue name prefix:

The prefix used at the start of temporary queues created by applications using this connection factory.

Required	No
Data type	String

Pass message payload by reference: When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION:
The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

Applications that use this connection factory to send messages must obey the following rules::

- The application does not modify the data object contained in a JMS object message.
- The application populates a JMS bytes message by using a single call to writeBytes(byte[]) and does not modify the byte array after it is set in the message.

When enabled, Object/Bytes Messages sent by a message producing application that has connected to the bus using this connection factory will not have their data copied when set and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.

Required	No
Data type	Boolean

Applications that use this connection factory to receive messages must obey the following rule::

- The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When enabled, Object Messages received by a message consuming application that has connected to this connection factory will only have their message data serialized by the system when absolutely necessary. The data obtained from these messages must be treated as readOnly by applications.

Required	No
Data type	Boolean

Log missing transaction contexts:

Whether or not the container logs that there is a missing transaction context when a connection is obtained.

The Java EE programming model indicates that connections should always have a transaction context. However, some applications do not correctly have a transaction context associated with them.

Select this property to log connections being created without a transaction context.

Required	No
Data type	Boolean

Manage cached handles:

Whether cached handles (handles held in instance variables in a bean) should be tracked by the container.

Select this option to track handle management, which can be useful for debugging purposes. However, tracking handles can significantly reduce performance if used at run time.

Required	No
Data type	Boolean

Share data source with CMP:

Allow sharing of connections between JMS and container-managed persistence (CMP) entity EJB beans.

This option is used as part of the task to enable container-managed persistence (CMP) entity beans to share the database connections used by the data store of a messaging engine. This has been estimated as a potential performance improvement of 15% for overall message throughput, but can only be used for entity beans connected to the application server that contains the messaging engine.

For more information about using this option, see the topic Enabling CMP entity beans and messaging engine data stores to share database connections.

Required	No
Data type	Boolean

XA recovery authentication alias:

The authentication alias used during XA recovery processing.

Select the alias to be used during transaction recovery processing.

This property provides a list of the JCA authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate during XA recovery processing.

If you have enabled security for the associated service integration bus, select the alias that specifies the user ID and password used for XA recovery that is valid in the user registry for WebSphere Application Server. This property must be set if bus security is enabled and XA transactions are to be used.

Required	No
Data type	drop-down list

Mapping-configuration alias:

Specifies the mapping configuration alias for the Java Authentication and Authorization Service (JAAS) mapping configuration that is used by this connection factory.

This field will be used only in the absence of a loginConfiguration on the component resource reference. The specification of a login configuration and associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is Container.If the DefaultPrincipalMapping login configuration is specified, the associated property will be a JAAS - J2C authentication data entry alias. See related item JAAS - J2C authentication data entry to define a new alias.

Required	No
Data type	drop-down list

Container-managed authentication alias:

This alias specifies a user ID and password to be used to authenticate connections to the JMS provider for container-managed authentication. This setting is only used when the res-auth value is container, and the authentication alias was not set when the application was deployed.

Required	No
Data type	drop-down list

Additional Properties

Connection pool properties
An optional set of connection pool settings.

Related Items

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

Buses
A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Default messaging provider queue [Settings]

A JMS queue is used as a destination for point-to-point messaging. Use JMS queue destination administrative objects to manage JMS queues for the default messaging provider.

To view this page in the console, click one of the following paths:

- **Resources -> JMS -> Queues -> *queue_name***
- **Resources -> JMS -> JMS providers -> *a_messaging_provider* -> [Additional Properties] Queues -> *queue_name***

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Scope:

Specifies the highest topological level at which application servers can use this resource object.

Required	No
Data type	String

Provider:

Specifies a JMS provider, which enables asynchronous messaging based on the Java Message Service (JMS). It provides J2EE connection factories to create connections for specific JMS queue or topic destinations. JMS provider administrative objects are used to manage JMS resources for the associated JMS provider.

Required	No
Data type	String

Name:

The required display name for the resource.

Required	Yes
Data type	String

JNDI name:

The JNDI name for the resource.

As a convention, use a JNDI name of the form jms/*Name*, where *Name* is the logical name of the resource. For more information about the use of JNDI and its syntax, see the topic JNDI support in WebSphere Application Server.

Required	Yes
Data type	String

Description:

An optional description for the resource.

Required No
Data type Text area

Bus name:

Enter the name of the bus on which the associated queue exists, or leave blank to use the bus to which the application connects.

Required No
Data type Custom

Queue name:

The name of the associated queue on the service integration bus.

Type the name of a queue that has been created on the service integration bus.

Required Yes
Data type Custom

Delivery mode:

The delivery mode for messages sent to this destination. This controls the persistence of messages on this destination.

Required No
Data type drop-down list
Range

Application
The persistence of messages on this topic is defined by the producing application.

Nonpersistent
All messages sent to this topic are treated as nonpersistent.

Persistent
All messages sent to this topic are treated as persistent.

Time to live:

The default length of time in milliseconds from its dispatch time that a message sent to this destination should be kept by the system.

Required No
Data type Integer
Range 0 through 574476389546486783

A value of 0 (zero) means that messages are kept indefinitely. The default for this property is null, which allows the application to determine the time to keep messages.

Priority:

The relative priority for messages sent to this destination, in the range 0 to 9, with 0 as the lowest priority and 9 as the highest priority.

If a value is not specified for this property, the message priority set by the producing application is used.

Required No
Data type Integer
Range 0 through 9

The message priority range is from 0 (lowest) through 9 (highest).

Read ahead:

Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

This property overrides the value set by the **Read ahead** property on the JMS connection factory.

Required No
Data type drop-down list
Range

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests for this destination.

Disabled
The messaging provider does not preemptively assign messages to consumers for this destination.

Inherit from connection factory
Read ahead optimization is defined on the connection.

Restrict messages to the local queue point if a queue point is configured on the connected messaging engine:

Indicates whether the underlying service integration bus queue destination is scoped to a local queue point when addressed by using this JMS queue. A local queue point is a queue point of the service integration bus queue that is configured on the messaging engine to which the JMS application is connected.

This option applies when using this JMS queue to send and receive messages, and when setting a reply queue in a request message. When a reply queue is set in a request message, the local queue point is on the messaging engine to which the application setting the reply queue is connected, not the messaging

engine to which the application that uses the reply queue sends the reply message. If the connected messaging engine does not have a queue point for the destination this option is ignored. The default value is FALSE.

This option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Required	Yes
Data type	Boolean
Default	FALSE

Local queue point preference:

Prefer to send messages to a local queue point

If a queue point for this queue exists on the messaging engine that the application is connected to, prefer to send all messages to it to improve performance. If this local queue point is not accepting new messages, the messages are sent to other queue points, if possible.

Required	No
Data type	Radio button

Do not prefer a local queue point over other queue points

All available queue points are treated equally, with no preference given to a local queue point. Messages are workload balanced across all queue points. This option is supported only when used by a JMS application that is running with a WebSphere Application ServerVersion 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application ServerVersion 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Required	No
Data type	Radio button

Message affinity across queue points:

Send all messages to the same queue point

A message producer created to use this queue sends all messages to the same queue point. This option does not influence the initial choice of queue point. This option applies to a message producer only if the queue is identified at the time the message producer is created, not at the time of sending messages. This option is supported only when used by a JMS application that is running with a WebSphere Application ServerVersion 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application ServerVersion 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Required	No
Data type	Radio button

Messages may be sent to different queue points

A message producer can send messages to different queue points, based on the availability of the queue points and workload balancing of the system.

Required	No
Data type	Radio button

Message visibility:

Only messages on a single queue point are visible

A consumer or browser has access to messages on only one queue point, not the entire queue. The single queue point is chosen by the messaging system. A queue point that is defined on the same messaging engine that the application is connected to is preferred to other queue points.

Required	No
Data type	Radio button

Messages on all queue points are visible

A consumer or browser has access to messages on all queue points of the queue. If you enable this option, all queue points are actively checked for messages. This means that no messages are left on an unattended queue point. However, there is a performance cost in scanning all queue points for messages, and you can no longer control the order in which messages are taken off the queue. This option is supported only when used by a JMS application that is running with a WebSphere Application ServerVersion 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application ServerVersion 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Required	No
Data type	Radio button

Related Items

Buses

A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Default messaging provider topic connection factory [Settings]

A JMS topic connection factory is used to create connections to the associated JMS provider of JMS topics, for publish/subscribe messaging. Use topic connection factory administrative objects to manage JMS topic connection factories for the default messaging provider.

To view this page in the console, click one of the following paths:

- **Resources -> JMS -> Topic connection factories -> *factory_name***
- **Resources -> JMS -> JMS providers -> *a_messaging_provider* -> [Additional Properties] Topic connection factories -> *factory_name***

Use this panel to browse or change the configuration properties of the selected JMS topic connection factory for use with the default messaging JMS provider. These configuration properties control how connections are created to associated JMS topics.

By default, connections created by using this JMS connection factory in the server containers (for example, from an enterprise bean) are pooled by using Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) connection pooling. You can modify the connection pool settings for this connection factory by selecting **Connection pool properties** in the Additional properties section of the administrative console panel.

The configuration of a container-managed authentication alias and mapping module on a connection factory are deprecated. You now set these properties in the bindings for the resource reference of the application. If you do not want to modify the bindings for an existing application, locate this connection factory in the J2C panels where you can still find these properties.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Scope:

Specifies the highest topological level at which application servers can use this resource object.

Required	No
Data type	String

Provider:

Specifies a JMS provider, which enables asynchronous messaging based on the Java Message Service (JMS). It provides J2EE connection factories to create connections for specific JMS queue or topic destinations. JMS provider administrative objects are used to manage JMS resources for the associated JMS provider.

Required	No
Data type	String

Name:

The required display name for the resource.

Required	Yes
Data type	String

JNDI name:

The JNDI name for the resource.

As a convention, use a JNDI name of the form jms/*Name*, where *Name* is the logical name of the resource. For more information about the use of JNDI and its syntax, see the topic JNDI support in WebSphere Application Server.

Required	Yes
Data type	String

Description:

An optional description for the resource.

Required	No
Data type	Text area

Category:

An optional category string to use when classifying or grouping the resource.

Required	No
Data type	String

Bus name:

The name of the bus to connect to.

This is the name of the service integration bus that this connection factory is used to create connections to.

Enter the name of the local bus in situations where an application makes connection to foreign buses.

Required	Yes
Data type	Custom

Target:

The name of a target that identifies a group of messaging engines. Specify the type of target using the Target type property.

This indicates the name of a target that is to be used to determine one or messaging engines to handle work. The type of target is indicated by the Target type property

Connections are load balanced across the available messaging engines that satisfy the selection criteria.

If want applications to be able to connect to any messaging engine in the bus, do not set this property.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required	No
Data type	String

Target type:

The type of target named in the Target property.

Required	No
Data type	drop-down list

Range

Bus member name
The name of a bus member. This option retrieves the active messaging engines that are hosted by the named bus member (an application server or server cluster).

To specify a non-clustered bus member the Target property must be set to `<Node01>.<server1>`, for example Node01.server1. For a cluster bus member the Target property must be set to the cluster name.

Custom messaging engine group name
The name of a custom group of messaging engines (that form a self-declaring cluster). This option retrieves the active messaging engines that have registered with the named custom group.

Messaging engine name
The name of a messaging engine. This option retrieves the available endpoints that can be used to reach the named messaging engine.

Target significance:

This property specifies the significance of the target group.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required No
Data type drop-down list
Range

Preferred
It is preferred that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, a messaging engine outside the target group is selected if available in the same service integration bus.

Required
It is required that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, the connection process fails.

Target inbound transport chain:

The name of the inbound transport chain that the application should target when connecting to a messaging engine in a separate process to the application. If a messaging engine in another process is chosen, a connection can be made only if the messaging engine is in a server that runs the specified inbound transport chain. Refer to the information center for more information.

If the selected messaging engine is in the same server as the application, a direct in-process connection is made and this transport chain property is ignored.

The transport chains represent network protocol stacks operating within a server. The name you specify must be one of the transport chains available in the server that hosts the messaging engine, as listed on the **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports** panel. The following transport chains are provided, but you can define your own transport chains on that panel.

InboundBasicMessaging
: This is a connection-oriented protocol that uses a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging
: This is the InboundBasicMessaging protocol wrapped in SSL.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required	No
Data type	String

Provider endpoints:

A comma-separated list of endpoint triplets, with the syntax hostName:portNumber:chainName, used to connect to a bootstrap server. For example Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging. If hostName is not specified, the default is localhost. If portNumber is not specified, the default is 7276. If chainName is not specified, the default is BootstrapBasicMessaging. Refer to the information center for more information.

You only have to modify this property if you have client applications running outside of an application server, or applications on a server in another cell, that want to use this connection factory to connect to the target service integration bus specified on the connection factory.

To use JMS destinations of the default messaging provider, an application connects to a messaging engine on the target service integration bus to which the destinations are assigned. For example, a JMS queue is assigned to a queue destination on a service integration bus.

Client applications running outside of an application server - for example, running in a client container or outside the WebSphere Application Server environment - cannot locate directly a suitable messaging engine to connect to in the target bus. Similarly, an application running on a server in one cell to connect to a target bus in another cell cannot locate directly a suitable messaging engine to connect to in the target bus.

In these scenarios, the clients (or servers in another bus) must complete a bootstrap process through a *bootstrap server* that is a member of the target bus. A bootstrap server is an application server running the SIB Service, but does not have to be running any messaging engines. The bootstrap server selects a messaging engine that is running in an application server that supports the required *target transport chain*. For the bootstrap process to be possible, you must configure one or more *provider end points* in the connection factory used by the client.

A bootstrap server uses a specific port and bootstrap transport chain. The port is the **SIB_ENDPOINT_ADDRESS** (or **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled), of the messaging engine that hosts the remote end of the link. Together with host name, these form the *endpoint address* of the bootstrap server.

The properties of a JMS connection factory used by an application control the selection of a suitable messaging engine and how the application connects to the selected messaging engine.

- If no security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7276 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used
- If security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7286 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used

Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

If you want an application to use a bootstrap server with a different endpoint address, you must specify the required endpoint address on the **Provider endpoints** property of the JMS connection factories that the client application uses. You can specify one or more endpoint addresses of bootstrap servers.

The endpoint addresses for bootstrap servers must be specified in every JMS connection factory that is used by applications outside of an application server. To avoid having to specify a long list of bootstrap servers, you can provide a few highly-available servers as dedicated bootstrap servers. Then you only have to specify a short list of bootstrap servers on each connection factory.

Note: When configuring a connection to a non-default bootstrap server, specify the required values for the endpoint address and use colons as separators.

For example: for a server assigned non-secure port 7278, on host boothost1, that uses the default transport chain BootstrapBasicMessaging:

```
boothost1:7278:BootstrapBasicMessaging
or
boothost1:7278
```

and for a server assigned secure port 7289, on host boothost2, that uses the predefined transport chain BootstrapTunneledSecureMessaging:

```
boothost2:7289:BootstrapTunneledSecureMessaging
```

The syntax for an endpoint address is as follows:

```
[ [host_name] [ ":" [port_number] [ ":" chain_name] ] ]
```

where:

host_name
: is the name of the host on which the server runs. It can be an IP address. For an IPv6 address, put square braces ([]) around *host_name* as shown in the example below:

  ```
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
  ```

  . If a value is not specified, the default is localhost.

Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

port_number
where specified, is one of the following addresses of the messaging engine that hosts the remote end of the link:

- **SIB_ENDPOINT_ADDRESS** if security is not enabled
- For secure connections, **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled.

If *port_number* is not specified, the default is 7276.

To find either of these values by using the administrative console, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Communications] Ports**.

chain_name
is the name of a predefined bootstrap transport chain used to connect to the bootstrap server. If not specified, the default is BootstrapBasicMessaging.

The following predefined bootstrap transport chains are provided:

BootstrapBasicMessaging
This corresponds to the server transport chain InboundBasicMessaging (JFAP-TCP/IP)

BootstrapSecureMessaging
This corresponds to the server transport chain InboundSecureMessaging (JFAP-SSL-TCP/IP)

BootstrapTunneledMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.) This transport chain tunnels JFAP and uses HTTP wrappers.

BootstrapTunneledSecureMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See**Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.) This transport chain tunnels JFAP and uses HTTP wrappers.

Specifying *host_name* : *chain_name* instead of *host_name* : : *chain_name* (with two colons) is incorrect. It is valid to enter nothing, or to enter any of the following: "a", "a:", ":7276", "::chain", and so on. The default value applies if you do not specify a value, but you must separate the fields with ":"s.

If you want to provide more than one bootstrap server, identify all the required endpoint addresses. Separate each endpoint address by a comma character. For example, to use the servers from the earlier example:

```
boothost1:7278:BootstrapBasicMessaging,
  boothost2:7289:BootstrapTunneledSecureMessaging,
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
```

Required	No
Data type	Text area

Connection proximity:

The proximity of messaging engines that can accept connection requests, in relation to the bootstrap messaging engine.

When a client issues a client connect request, the processing attaches to the required bus according to the following logic:

- If a target group is specified, connect to the first messaging engine that satisfies the following conditions for the target type:
  - **Server** Look for a messaging engine in the same server.
  - **Cluster** Look for a messaging engine in the same server, then on other servers in the same cluster.
  - **Host** Look for a messaging engine in the same server, then on other servers in the same cluster, then on other servers in the same host.
  - **Bus** Look for a messaging engine in the same server, then on other servers in the same cluster, then on other servers in same host, then any other messaging engine on the same bus.
- If a target group is not specified, or a target group is specified but no suitable messaging engine is found and target significance is Preferred, connect to the first messaging engine that satisfies the following conditions for the target type:
  - **Server** Look for a messaging engine in the same server.
  - **Cluster** Connection fails.
  - **Host** Look for a messaging engine in the same server, then on other servers in the same host.
  - **Bus** Look for a messaging engine in the target group in same server, then on other servers in same host, then any other messaging engine on the same bus.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

Required No
Data type drop-down list
Range

Bus Connections can be made to messaging engines in the same bus.

Cluster Connections can be made to messaging engines in the same server cluster.

Host Connections can be made to messaging engines in the same host.

Server Connections can be made to messaging engines in the same application server.

Client identifier:

The JMS client identifier needed for durable topic subscriptions on all connections created using this connection factory.

Required No
Data type String

Durable subscription home:

The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS connection factory.

To enable applications to use durable subscriptions, you must set this property.

Required No
Data type Custom

Nonpersistent message reliability:

The reliability applied to nonpersistent JMS messages sent using this connection factory.

You can change the delivery reliability option for the destination of a message that is sent by a JMS application as Nonpersistent. The default is Express nonpersistent but you have a range of other options, including those with persistent characteristics, with Assured persistent being the most reliable. For more information see the topic Message reliability levels - JMS delivery mode and service integration quality of service.

Required
No

Data type
drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

As bus destination
Use the delivery option configured for the bus destination.

Persistent message reliability:

The reliability applied to persistent JMS messages sent using this connection factory.

You can change the delivery reliability option for the destination of a message that is sent by a JMS application as Persistent. The default is Reliable persistent but you have a range of other options including those with nonpersistent characteristics, with Best effort nonpersistent being the least reliable. For more information see the topic Message reliability levels - JMS delivery mode and service integration quality of service.

Important: If you change the delivery reliability options of a message sent by a JMS application from one of the **Persistent message reliability** options (Assured persistent and Reliable persistent) to one of the **Nonpersistent message reliability** options (Best effort nonpersistent, Express nonpersistent, and Reliable nonpersistent), you risk losing messages in certain circumstances. For example, at server restart, or when there is heavy workload.

Required
No

Data type
drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

As bus destination
Use the delivery option configured for the bus destination.

Read ahead:

Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

You can override this property for individual JMS destinations by setting the **Read ahead** property on the JMS destination.

Required
No

Data type
drop-down list

Range

Default The message provider preemptively assigns messages to consumers on nondurable subscriptions and unshared durable subscriptions. That is, read ahead optimization is turned on only when there can only be a single consumer.

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Disabled
The messaging provider does not preemptively assign messages to consumers.

Temporary topic name prefix:

The prefix used at the start of temporary topics created by applications using this connection factory.

Required No
Data type String

Share durable subscriptions:

Controls whether or not durable subscriptions are shared across connections with members of a server cluster.

Usually, only one session at a time can have a TopicSubscriber for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers.

Required No
Data type drop-down list
Range

In cluster
Allows sharing of durable subscriptions when connections are made from within a server cluster.

Always shared
Durable subscriptions can be shared across connections.

Never shared
Durable subscriptions are never shared across connections.

Pass message payload by reference: When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION:
The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

Applications that use this connection factory to send messages must obey the following rules::

- The application does not modify the data object contained in a JMS object message.
- The application populates a JMS bytes message by using a single call to writeBytes(byte[]) and does not modify the byte array after it is set in the message.

When enabled, Object/Bytes Messages sent by a message producing application that has connected to the bus using this connection factory will not have their data copied when set and the system will only serialize the message data when absolutely necessary. Applications sending such messages must not modify the data once it has been set into the message.

Required	No
Data type	Boolean

Applications that use this connection factory to receive messages must obey the following rule::

- The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When enabled, Object Messages received by a message consuming application that has connected to this connection factory will only have their message data serialized by the system when absolutely necessary. The data obtained from these messages must be treated as readOnly by applications.

Required	No
Data type	Boolean

Log missing transaction contexts:

Whether or not the container logs that there is a missing transaction context when a connection is obtained.

The Java EE programming model indicates that connections should always have a transaction context. However, some applications do not correctly have a transaction context associated with them.

Select this property to log connections being created without a transaction context.

Required	No
Data type	Boolean

Manage cached handles:

Whether cached handles (handles held in instance variables in a bean) should be tracked by the container.

Select this option to track handle management, which can be useful for debugging purposes. However, tracking handles can significantly reduce performance if used at run time.

Required	No
Data type	Boolean

Share data source with CMP:

Allow sharing of connections between JMS and container-managed persistence (CMP) entity EJB beans.

This option is used as part of the task to enable container-managed persistence (CMP) entity beans to share the database connections used by the data store of a messaging engine. This has been estimated as a potential performance improvement of 15% for overall message throughput, but can only be used for entity beans connected to the application server that contains the messaging engine.

For more information about using this option, see the topic Enabling CMP entity beans and messaging engine data stores to share database connections.

Required	No
Data type	Boolean

XA recovery authentication alias:

The authentication alias used during XA recovery processing.

Select the alias to be used during transaction recovery processing.

This property provides a list of the J2C authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate during XA recovery processing.

If you have enabled security for the associated service integration bus, select the alias that specifies the user ID and password used for XA recovery that is valid in the user registry for WebSphere Application Server. This property must be set if bus security is enabled and XA transactions are to be used.

Required	No
Data type	drop-down list

Mapping-configuration alias:

Specifies the mapping configuration alias for the Java Authentication and Authorization Service (JAAS) mapping configuration that is used by this connection factory.

This field will be used only in the absence of a loginConfiguration on the component resource reference. The specification of a login configuration and associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is Container.If the DefaultPrincipalMapping login configuration is specified, the associated property will be a JAAS - J2C authentication data entry alias. See related item JAAS - J2C authentication data entry to define a new alias.

Required	No
Data type	drop-down list

Container-managed authentication alias:

This alias specifies a user ID and password to be used to authenticate connections to the JMS provider for container-managed authentication. This setting is only used when the res-auth value is container, and the authentication alias was not set when the application was deployed.

Required	No

Data type drop-down list

Additional Properties

Connection pool properties
An optional set of connection pool settings.

Related Items

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

Buses
A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Default messaging provider topic [Settings]

A JMS topic is used as a destination for publish/subscribe messaging. Use topic destination administrative objects to manage JMS topics for the default messaging provider.

To view this page in the console, click one of the following paths:

- **Resources -> JMS -> Topics -> *topic_name***
- **Resources -> JMS -> JMS providers -> *a_messaging_provider* -> [Additional Properties] Topics** > ***topic_name***

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Scope:

Specifies the highest topological level at which application servers can use this resource object.

Required No
Data type String

Provider:

Specifies a JMS provider, which enables asynchronous messaging based on the Java Message Service (JMS). It provides J2EE connection factories to create connections for specific JMS queue or topic destinations. JMS provider administrative objects are used to manage JMS resources for the associated JMS provider.

Required No
Data type String

Name:

The required display name for the resource.

Required Yes
Data type String

JNDI name:

The JNDI name for the resource.

As a convention, use a JNDI name of the form jms/*Name*, where *Name* is the logical name of the resource. For more information about the use of JNDI and its syntax, see the topic JNDI support in WebSphere Application Server.

Required Yes
Data type String

Description:

An optional description for the resource.

Required No
Data type Text area

Topic name:

The name of the topic that this JMS topic is assigned to, in the topic space defined by the Topic space property.

To specify a multi-level topic name, separate each level by a forward slash character (/); for example:
`schools/eton/subjects/maths`

Required No
Data type String

Bus name:

Name of bus hosting topic.

Required No
Data type Custom

Topic space:

The name of the topic space that contains the topic, on the service integration bus defined by the Bus name property

Type the name of a topic space that has been created on the service integration bus.

Required Yes
Data type Custom

JMS delivery mode:

The delivery mode for messages sent to this destination. This controls the persistence of messages on this destination.

Required No
Data type drop-down list
Range

Application
The persistence of messages on this topic is defined by the producing application.

Nonpersistent
All messages sent to this topic are treated as nonpersistent.

Persistent
All messages sent to this topic are treated as persistent.

Time to live:

The default length of time in milliseconds from its dispatch time that a message sent to this destination should be kept by the system.

Messages are deleted after the specified time. If a value is not specified for this property, the time limit set by the producing application is used.

Required No
Data type Long
Range 0 through 574476389546486783

A value of 0 (zero) means that messages are kept indefinitely. The default for this property is null, which allows the application to determine the time to keep messages.

Message priority:

The relative priority for messages sent to this destination, in the range 0 to 9, with 0 as the lowest priority and 9 as the highest priority.

If a value is not specified for this property, the message priority set by the producing application is used.

Required No
Data type Integer
Range 0 through 9

The message priority range is from 0 (lowest) through 9 (highest).

Read ahead:

Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

This property overrides the value set by the **Read ahead** property on the JMS connection factory.

Required
No

Data type
drop-down list

Range

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests for this destination.

Disabled
The messaging provider does not preemptively assign messages to consumers for this destination.

Inherit from connection factory
Read ahead optimization is defined on the connection.

Related Items

Buses
A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

SIBJMSAdminCommands command group for the AdminTask object

You can use these administrative commands to manage JMS resources for the default messaging provider.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

createSIBJMSActivationSpec command

Use the createSIBJMSActivationSpec command to create a new JMS activation specification for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The createSIBJMSActivationSpec command creates a new JMS activation specification at a specific scope.

Target object

The scope of the default messaging provider at which the JMS activation specification is to be created.

Required parameters

-name
The administrative name assigned to this activation specification.

-jndiName
The JNDI name that is specified in the bindings for message-driven beans associated with this activation specification.

-destinationJndiName
The JNDI name of the destination JMS queue or topic used by the message-driven bean.

Optional parameters

-description
An optional description for the activation specification.

-destinationType
An option to determine whether the message_driven bean uses a JMS queue or a JMS topic. Select one of the following values:

Queue
The message-driven bean uses a JMS queue. The JNDI name of the JMS queue is specified on the **Destination JNDI name** property.

Topic The message-driven bean uses a JMS topic. The JNDI name of the JMS topic is specified on the **Destination JNDI name** property.

-messageSelector
The JMS message selector used to determine which messages the message-driven bean receives. The value is a string that is used to select a subset of the available messages. The syntax is based on a subset of the SQL 92 conditional expression syntax, as described in the JMS specification. Refer to the information center for more information.

The selector string can refer to fields in the JMS message header and fields in the message properties. Message selectors cannot reference message body values.

A null value (an empty string) indicates that there is no message selector for the message consumer.

-busName
The name of the service integration bus to which connections are made. This must be the name of the bus on which the destination identified by the **-destinationJndiName** property is defined.

-acknowledgeMode
The acknowledge mode indicates how a message received by a message-driven bean should be acknowledged. Select one of the following values:

Auto-acknowledge
The session automatically acknowledges the delivery of a message.

Duplicates-ok auto-acknowledge
The session lazily acknowledges the delivery of messages, which can improve performance, but can lead to a message-driven bean receiving a message more than once.

-target
The name of a target that identifies a group of messaging engines. Specify the type of target by using the **Target type** property.

-targetType
The type of target named in the **-target** property. Select one of the following values:

Bus member name
The name of a bus member. This option retrieves the active messaging engines that are hosted by the named bus member (an application server or server cluster).

Custom messaging engine group name
The name of a custom group of messaging engines (that form a self-declaring cluster). This option retrieves the active messaging engines that have registered with the named custom group.

Messaging engine name
The name of a messaging engine. This option retrieves the available endpoints that can be used to reach the named messaging engine.

-targetSignificance
This property specifies the significance of the target group. Select one of the following values:

Preferred
It is preferred that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, a messaging engine outside the target group is selected if available in the same service integration bus.

Required
It is required that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, the connection process fails.

-targetTransportChain
The name of the inbound transport chain that the application should target when connecting to a messaging engine in a separate process to the application. If a messaging engine in another process

is chosen, a connection can be made only if the messaging engine is in a server that runs the specified inbound transport chain. Refer to the information center for more information.

If the selected messaging engine is in the same server as the application, a direct in-process connection is made and this transport chain property is ignored.

The transport chains represent network protocol stacks operating within a server. The name you specify must be one of the transport chains available in the server that hosts the messaging engine, as listed on the **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports** panel. The following transport chains are provided, but you can define your own transport chains on that panel.

InboundBasicMessaging
This is a connection-oriented protocol that uses a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging
This is the InboundBasicMessaging protocol wrapped in SSL.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

-providerEndPoints
A comma-separated list of endpoint triplets, with the syntax *host_name*:*port_number*:*chain_name*, that is used to connect to a bootstrap server. For example `Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging`.

Provider endpoints are used if the specified bus cannot be found in the local cell. Message-driven bean applications first attempt to connect the specified bus in the local cell. If this attempt fails, provider endpoints are used to allow the applications to consume messages from a remote cell.

If the host name is not specified, localhost is used as a default value.

If the port number is not specified, 7276 is used as the default value.

If the protocol is not specified, a predefined chain such as BootstrapBasicMessaging is used as the default value.

-authenticationAlias
The name of a J2C authentication alias used for component-managed authentication of connections to the service integration bus.

A Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) authentication alias specifies the user ID and password that is used to authenticate the creation of a new connection to the JMS provider.

-maxBatchSize
The maximum number of messages in a single batch delivered serially to a single message-driven bean instance. Batching of messages can improve performance particularly when used with Acknowledge mode set to Duplicates-ok auto-acknowledge. If message ordering must be retained across failed deliveries, set the batch size to 1.

-maxConcurrency
The maximum number of endpoints to which messages are delivered concurrently.

Increasing this number can improve performance but can increase the number of threads that are in use at any one time. If message ordering must be retained across failed deliveries, set the maximum

concurrent endpoints to 1. Message ordering applies only if the destination that the message-driven bean is consuming from is not a partitioned destination. Partitioned destinations are used in a workload sharing scenario in a cluster.

-subscriptionDurability
This option specifies whether a JMS topic subscription is durable or nondurable.

Usually, only one application at a time can have a consumer for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers. Select one of the following values:

Durable
The messaging provider stores messages while the message-driven bean is not available, and delivers the messages when the message-driven bean becomes available again.

Nondurable
The messaging provider does not store and redeliver messages if a message-driven bean is not available.

-subscriptionName
The subscription name needed for durable topic subscriptions. Required field when using a durable topic subscription.

Each JMS durable subscription is identified by a subscription name (specified on this property). A JMS connection also has an associated client identifier (specified on the Client identifier property), which is used to associate a connection and its objects with the list of messages (on the durable subscription) that is maintained by the JMS provider for the client.

This subscription name must be unique within a given client identifier.

-clientId
The JMS client identifier needed for durable topic subscriptions on all connections created using this activation specification.

The value specified is a unique identifier for a client (message-driven bean). The client identifier is used to associate a client connection with the list of messages (on a durable subscription) that the messaging provider keeps for the client. When a client becomes available, after it has been unavailable, the messaging provider uses the client identifier to re-deliver stored messages to the correct client.

-durableSubscriptionHome
The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS activation specification. This is a required field when using a durable topic subscription.

Administrators can manage the runtime state of durable subscriptions through publication points for this messaging engine.

-shareDurableSubscriptions
Controls whether or not durable subscriptions are shared across connections with members of a server cluster.

Usually, only one session at a time can have a TopicSubscriber for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers, one on each application server in the server cluster.

This option should be changed from its default only in WebSphere Application Server environments that support server clusters.

Select one of the following values:

In cluster
Allows sharing of durable subscriptions when connections are made from within a server cluster.

Always shared
Durable subscriptions can be shared across connections.

Never shared
Durable subscriptions are never shared across connections.

-shareDataSourceWithCmp
Allow sharing of connections between JMS and container-managed persistence (CMP) entity beans.

True | False

This option is used as part of the task to enable container-managed persistence (CMP) entity beans to share the database connections used by the data store of a messaging engine. This has been estimated as a potential performance improvement of 15% for overall message throughput, but can only be used for entity beans connected to the application server that contains the messaging engine.

For more information about using this option, see "Enabling CMP entity beans and messaging engine data stores to share database connections" on page 530.

-readAhead
Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

You can override this property for individual JMS destinations by setting the **Read ahead** property on the JMS destination.

Select one of the following values:

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Disabled
The messaging provider does not preemptively assign messages to consumers.

Default
The message provider preemptively assigns messages to consumers on nondurable subscriptions and unshared durable subscriptions. That is, read ahead optimization is turned on only when there can only be a single consumer.

The "pass message payload by reference" properties:

-forwarderDoesNotModifyPayloadAfterSet
true | false (default false)

Applications resending messages that were originally received using this activation specification must obey the following rules:

- The application can replace the data object in a JMS object message, provided that the data object has not yet been set in the message. The application does not modify or replace the data object after it is set in the message.
- The application can replace the byte array in a JMS bytes message, but only by using a single call to writeBytes(byte[]), and provided that the byte array has not yet been set in the message. The application does not modify or replace the byte array after it is set in the message.

-consumerDoesNotModifyPayloadAfterGet
true | false (default false)

Applications that use this activation specification to receive messages must obey the following rule: The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION: The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

-alwaysActivateAllMDBs
True | False

This property is only used when the MDB application is running on a server that is a member of the bus that the application is targeting. It has no effect when the MDB is running on a server that is not a member of the target bus.

If the MDB application is running on a server that is a member of the target bus, enabling this option allows the MDB application to process messages whether or not the server also hosts a running messaging engine. If this option is not enabled, then MDB applications on servers that do not have a local ME running do not process messages.

-retryInterval
The delay (in seconds) between attempts to connect to a messaging engine, both for the initial connection, and any subsequent attempts to establish a better connection.

-userName
The user identity for Java 2 connector security to use.

-password
The password for Java 2 connector security to use.

-WAS_EndpointInitialState
ACTIVE |INACTIVE

This property determines whether the endpoint is activated when the endpoint is registered. If the property is set to active, message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.

Example

- The following example shows an activation specification being created using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.createSIBJMSActivationSpec("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml)", ["-name", "myjmsas", "-jndiName", "jms/myjmsas",
 "-destinationJndiName", "jms/mqueue", "-busName", "abus"])
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)'

wsadmin>AdminTask.listSIBJMSActivationSpecs("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml)")
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)'
```

- The following example shows an activation specification being created using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createSIBJMSActivationSpec
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name myjmsas -jndiName jms/myjmsas -destinationJndiName jms/mqueue -busName
```

```
abus} myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)

wsadmin>$AdminTask listSIBJMSActivationSpecs 9994GKCNode01
(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
```

- The following example creates an activation specification with the WAS_EndpointInitialState optional parameter activated, using Jython:

```
wsadmin>attrs = '[[name "WAS_EndpointInitialState"] [required "false"] [type "java.lang.String"] [value "ACTIVE"]]'
wsadmin>AdminConfig.getid("/Node:myNode01")
myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)'
wsadmin>theActSpec = AdminTask.createSIBJMSActivationSpec("myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)",
        '-name testas -jndiName testas -destinationJndiName testq -destinationType javax.jms.Queue')
wsadmin>AdminConfig.create('J2EEResourceProperty',theActSpec, attrs)
'WAS_EndpointInitialState(cells/myCell01/nodes/myNode01|resources.xml#J2EEResourceProperty_1298546239332)'
```

deleteSIBJMSActivationSpec command

Use the deleteSIBJMSActivationSpec command to delete a JMS activation specification for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('SIBJMSAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('`*command_name*`')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes the specified JMS activation specification.

Target object

A JMS activation specification.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminTask.listSIBJMSActivationSpecs("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)")
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
anojmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729538669)
test(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729811201)'

wsadmin>AdminTask.deleteSIBJMSActivationSpec("anojmsas(cells/
9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CActivationSpec_1098729538669)")
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729538669)'

wsadmin>AdminTask.listSIBJMSActivationSpecs("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)")
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
test(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729811201)'
```

- Using Jacl:

```
wsadmin>$AdminTask listSIBJMSActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
anojmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729538669)
test(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729811201)

wsadmin>$AdminTask deleteSIBJMSActivationSpec
anojmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729538669)
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729538669)

wsadmin>$AdminTask listSIBJMSActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
test(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098729811201)
```

listSIBJMSActivationSpecs command

Use the listSIBJMSActivationSpecs command to list all JMS activation specifications for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all JMS activation specifications for the default messaging provider at a specific scope.

Target object

Scope of the default messaging provider at which the JMS activation specifications were created.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.listSIBJMSActivationSpecs("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)")
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listSIBJMSActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
```

modifySIBJMSActivationSpec command

Use the modifySIBJMSActivationSpec command to change properties of a JMS activation specification for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command changes the properties of a JMS activation specification.

Target object

The scope of the default messaging provider at which the JMS activation specification is to be modified.

Required parameters

None.

Optional parameters

-name
The administrative name assigned to this activation specification.

-jndiName
The JNDI name that is specified in the bindings for message-driven beans associated with this activation specification.

-description
An optional description for the activation specification.

-destinationType
Use this parameter to determine whether the message-driven bean uses a JMS queue or a JMS topic. Select one of the following values:

Queue
The message-driven bean uses a JMS queue. The JNDI name of the JMS queue is specified on the **Destination JNDI name** property.

Topic The message-driven bean uses a JMS topic. The JNDI name of the JMS topic is specified on the **Destination JNDI name** property.

-destinationJndiName
The JNDI name of the destination JMS queue or topic used by the message-driven bean.

-messageSelector
string

-busName
Enter the name of the service integration bus to which connections are made. This must be the name of the bus on which the destination identified by the **-destinationJndiName** property is defined.

-acknowledgeMode
The acknowledge mode indicates how a message received by a message-driven bean should be acknowledged. Select one of the following values:

Auto-acknowledge
The session automatically acknowledges the delivery of a message.

Duplicates-ok auto-acknowledge
The session lazily acknowledges the delivery of messages, which can improve performance, but can lead to a message-driven bean receiving a message more than once.

-target
The name of a target that identifies a group of messaging engines. Specify the type of target using the Target type property.

-targetType
The type of target named in the **-target** property. Select one of the following values:

Bus member name
The name of a bus member. This option retrieves the active messaging engines that are hosted by the named bus member (an application server or server cluster).

Custom messaging engine group name
The name of a custom group of messaging engines (that form a self-declaring cluster). This option retrieves the active messaging engines that have registered with the named custom group.

Messaging engine name
The name of a messaging engine. This option retrieves the available endpoints that can be used to reach the named messaging engine.

-targetSignificance
This property specifies the significance of the target group. Select one of the following values:

Preferred
It is preferred that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, a messaging engine outside the target group is selected if available in the same service integration bus.

Required
It is required that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, the connection process fails.

-targetTransportChain
The name of the inbound transport chain that the application should target when connecting to a messaging engine in a separate process to the application. If a messaging engine in another process is chosen, a connection can be made only if the messaging engine is in a server that runs the specified inbound transport chain. Refer to the information center for more information.

If the selected messaging engine is in the same server as the application, a direct in-process connection is made and this transport chain property is ignored.

The transport chains represent network protocol stacks operating within a server. The name you specify must be one of the transport chains available in the server that hosts the messaging engine, as listed on the **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports** panel. The following transport chains are provided, but you can define your own transport chains on that panel.

InboundBasicMessaging
This is a connection-oriented protocol that uses a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging
This is the InboundBasicMessaging protocol wrapped in SSL.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

-providerEndPoints
A comma-separated list of endpoint triplets, with the syntax *host_name*:*port_number*:*chain_name*, used to connect to a bootstrap server. For example
`Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging`.

Provider endpoints are not used unless the specified bus cannot be found in the local cell. MDB applications first attempt to connect the specified bus in the local cell. If this attempt fails, provider endpoints are used to allow the applications to consume messages from a remote cell.

If the host name is not specified, localhost is used as a default value.

If the port number is not specified, 7276 is used as the default value.

If the protocol is not specified, a predefined chain such as BootstrapBasicMessaging is used as the default value.

-authenticationAlias
The name of a J2C authentication alias used for component-managed authentication of connections to the service integration bus.

A Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) authentication alias specifies the user ID and password that is used to authenticate the creation of a new connection to the JMS provider.

-maxBatchSize
The maximum number of messages in a single batch delivered serially to a single message-driven bean instance. Batching of messages can improve performance particularly when used with Acknowledge mode set to Duplicates-ok auto-acknowledge. If message-ordering must be retained across failed deliveries, set the batch size to 1

-maxConcurrency
The maximum number of endpoints to which messages are delivered concurrently.

Increasing this number can improve performance but can increase the number of threads that are in use at any one time. If message ordering must be retained across failed deliveries, set the maximum concurrent endpoints to 1. Message ordering applies only if the destination that the message-driven bean is consuming from is not a partitioned destination. Partitioned destinations are used in a workload sharing scenario in a cluster.

-subscriptionDurability
Whether a JMS topic subscription is durable or nondurable

Usually, only one application at a time can have a consumer for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers. Select one of the following values:

Durable
The messaging provider stores messages while the message-driven bean is not available, and delivers the messages when the message-driven bean becomes available again.

Nondurable
The messaging provider does not store and redeliver messages if a message-driven bean is not available.

-subscriptionName
The subscription name needed for durable topic subscriptions. Required field when using a durable topic subscription.

Each JMS durable subscription is identified by a subscription name (specified on this property). A JMS connection also has an associated client identifier (specified on the Client identifier property), which is used to associate a connection and its objects with the list of messages (on the durable subscription) that is maintained by the JMS provider for the client.

This subscription name must be unique within a given client identifier.

-clientId
The JMS client identifier needed for durable topic subscriptions on all connections created using this activation specification.

The value specified is a unique identifier for a client (message-driven bean). The client identifier is used to associate a client connection with the list of messages (on a durable subscription) that the messaging provider keeps for the client. When a client becomes available again, after a being unavailable, the messaging provider uses the client identifier to redeliver stored messages to the correct client.

-durableSubscriptionHome
The name of the messaging engine used to store messages delivered to durable subscriptions for objects created from this JMS activation specification. This is a required field when using a durable topic subscription.

Administrators can manage the runtime state of durable subscriptions through publication points for this messaging engine.

-shareDurableSubscriptions
Controls whether or not durable subscriptions are shared across connections with members of a server cluster.

Usually, only one session at a time can have a TopicSubscriber for a particular durable subscription. This property enables you to override this behavior, to enable a durable subscription to have multiple simultaneous consumers, one on each application server in the server cluster.

This option should be changed from its default only in WebSphere Application Server environments that support server clusters.

Select one of the following values:

In cluster
Allows sharing of durable subscriptions when connections are made from within a server cluster.

Always shared
Durable subscriptions can be shared across connections.

Never shared
Durable subscriptions are never shared across connections.

-shareDataSourceWithCmp
Allow sharing of connections between JMS and container-managed persistence (CMP) entity beans.

True | False

This option is used as part of the task to enable container-managed persistence (CMP) entity beans to share the database connections used by the data store of a messaging engine. This has been estimated as a potential performance improvement of 15% for overall message throughput, but can only be used for entity beans connected to the application server that contains the messaging engine.

For more information about using this option, see Enabling CMP entity beans and messaging engine data stores to share database connections..

-readAhead
Read ahead is an optimization that preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Messages that are assigned to a consumer are locked on the server and cannot be consumed by any other consumers for that destination. Messages that are assigned to a consumer, but not consumed before that consumer is closed, are subsequently unlocked on the server and then available for receipt by other consumers.

You can override this property for individual JMS destinations by setting the **Read ahead** property on the JMS destination.

Select one of the following values:

Enabled
The messaging provider preemptively assigns messages to consumers. This improves the time taken to satisfy consumer requests.

Disabled
The messaging provider does not preemptively assign messages to consumers.

Default
The message provider preemptively assigns messages to consumers on nondurable subscriptions and unshared durable subscriptions. That is, read ahead optimization is turned on only when there can only be a single consumer.

The “pass message payload by reference” properties:

-forwarderDoesNotModifyPayloadAfterSet
true | false (default false)

Applications resending messages that were originally received using this activation specification must obey the following rules:

- The application can replace the data object in a JMS object message, provided that the data object has not yet been set in the message. The application does not modify or replace the data object after it is set in the message.
- The application can replace the byte array in a JMS bytes message, but only by using a single call to writeBytes(byte[]), and provided that the byte array has not yet been set in the message. The application does not modify or replace the byte array after it is set in the message.

-consumerDoesNotModifyPayloadAfterGet
true | false (default false)

Applications that use this activation specification to receive messages must obey the following rule: The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION: The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

-alwaysActivateAllMDBs
True | False

This property is only used when the MDB application is running on a server that is a member of the bus that the application is targeting. It has no effect when the MDB is running on a server that is not a member of the target bus.

If the MDB application is running on a server that is a member of the target bus, enabling this option allows the MDB application to process messages whether or not the server also hosts a running messaging engine. If this option is not enabled, then MDB applications on servers that do not have a local ME running do not process messages.

-retryInterval
The delay (in seconds) between attempts to connect to a messaging engine, both for the initial connection, and any subsequent attempts to establish a better connection.

-userName
The user identity for Java 2 connector security to use.

-password
The password for Java 2 connector security to use.

-WAS_EndpointInitialState
This property determines whether the endpoint is activated when the endpoint is registered. If the property is set to active, message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.

The value of this parameter must be ACTIVE or INACTIVE.

Example

- The following example shows an activation specification being modified using Jython:

```
wsadmin>AdminTask.modifySIBJMSActivationSpec("myjmsas(cells/
9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CActivationSpec_1098726667851)",
 ["-jndiName", "jms/jmsas4q1",
 "-description", "JMS activation specification for myqueue1",
 "-destinationJndiName", "jms/myqueue1"])
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivation Spec_1098726667851)'
```

- The following example shows an activation specification being modified using Jacl:

```
wsadmin>$AdminTask modifySIBJMSActivationSpec
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
{-jndiName jms/jmsas4q1 -description "JMS activation specification
for myqueue1" -destinationJndiName jms/myqueue1}
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivation Spec_1098726667851)
```

- The following example modifies an activation specification by activating the WAS_EndpointInitialState optional parameter, using Jython:

```
wsadmin>attrs = '[[name "WAS_EndpointInitialState"] [required "false"] [type "java.lang.String"] [value "ACTIVE"]]'
wsadmin>AdminConfig.getid("/Node:myNode01")
'myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)'
wsadmin>AdminTask.listSIBJMSActivationSpecs("myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)")
'newas(cells/myCell01/nodes/myNode01|resources.xml#J2CActivationSpec_1298546034140)'
wsadmin>AdminConfig.create('J2EEResourceProperty',
"testas(cells/myCell01/nodes/myNode01|resources.xml#J2CActivationSpec_1298546034140)", attrs)
'WAS_EndpointInitialState(cells/myCell01/nodes/myNode01|resources.xml#J2EEResourceProperty_1298546239332)'
```

showSIBJMSActivationSpec command

Use the showSIBJMSActivationSpec command to show properties of a JMS activation specification for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('SIBJMSAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('command_name')`

Purpose

This command returns a set of property-value pairs for the specified JMS activation specification.

Target object

A JMS activation specification.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.listSIBJMSActivationSpecs("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)")
'myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)'

wsadmin>AdminTask.showSIBJMSActivationSpec("myjmsas(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#J2CActivationSpec_1098726667851)")
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
'{busName=abus, subscriptionDurability=Nondurable, description=,
destinationType=, password=, targetTransportChain=,
acknowledgeMode=Auto-acknowledge, readAhead=Default, clientId=,
authenticationAlias=, name=myjmsas, maxConcurrency=10, maxBatchSize=1,
durableSubscriptionHome=, userName=, messageSelector=,
shareDurableSubscriptions=InCluster, jndiName=jms/myjmsas,
shareDataSourceWithCMP=false, destination=, destinationJndiName=jms/mqueue,
subscriptionName=}'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listSIBJMSActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)

wsadmin>$AdminTask showSIBJMSActivationSpec
myjmsas(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098726667851)
{busName=abus, subscriptionDurability=Nondurable, description=,
destinationType=, password=, targetTransportChain=,
acknowledgeMode=Auto-acknowledge, readAhead=Default, clientId=,
authenticationAlias=, name=myjmsas, maxConcurrency=10, maxBatchSize=1,
durableSubscriptionHome=, userName=, messageSelector=,
shareDurableSubscriptions=InCluster, jndiName=jms/myjmsas,
shareDataSourceWithCMP=false, destination=, destinationJndiName=jms/mqueue,
subscriptionName=}
```

createSIBJMSConnectionFactory command

Use the createSIBJMSConnectionFactory command to create a new JMS connection factory for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a new JMS connection factory at a specific scope.

Target object

Scope of the default messaging provider at which the JMS connection factory is to be created.

Required parameters

-name
The administrative name assigned to this connection factory.

-jndiName
The JNDI name that is specified in the bindings for message-driven beans associated with this connection factory.

-busName
Enter the name of the service integration bus to which connections are made. This must be the name of the bus on which the destination identified by the **-destinationJndiName** property is defined.

Optional parameters

-type
queue | topic

The **type** parameter is used to specify the type of connection factory to create. To create a queue connection factory, set this parameter to queue. To create a topic connection factory, set this parameter to topic. Leave this parameter unset to create a generic connection factory.

-category
An optional category string to use when classifying or grouping the resource.

-description
text

-logMissingTransactionContext
True | False

-manageCachedHandles
True | False

-clientID
id

-userName
name

-password
password

-target
The name of a target that identifies a group of messaging engines. Specify the type of target using the Target type property.

Before the connection proximity search is performed to select a suitable messaging engine, the set of messaging engines that are members of the specified target group are selected. The connection proximity search is then restricted to these messaging engines. If a target group is not specified (the default), then all messaging engines in the bus are considered during the connection proximity search. For example, if the Target type property is set to Bus member name, the Target property specifies the name of the bus member from which suitable messaging engines can be chosen.

-targetType
The type of target named in the Target property.

Select one of the following values:

Bus member name
The name of a bus member. This option retrieves the active messaging engines that are hosted by the named bus member (an application server or server cluster).

To specify a non-clustered bus member the **-target** property must be set to *node_name*.*server_name*, for example `Node01.server1`. For a cluster bus member the **-target** property must be set to the cluster name.

Custom messaging engine group name
The name of a custom group of messaging engines (that form a self-declaring cluster). This option retrieves the active messaging engines that have registered with the named custom group.

Messaging engine name
The name of a messaging engine. This option retrieves the available endpoints that can be used to reach the named messaging engine.

-targetSignificance
This property specifies the significance of the target group.

This property defines whether the connection proximity search is restricted to only the messaging engines in the target group.

Select one of the following values:

Preferred
It is preferred that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, a messaging engine outside the target group is selected if available in the same service integration bus.

Note: A connection to a non-preferred target might be returned even if a preferred one is available. This can happen when connection pooling is enabled for a ConnectionFactory, which it is by default when you use a JMS ConnectionFactory in a server environment:

- When a preferred messaging engine is not available, a connection to a non-preferred one can be created and stored in the connection pool.
- The next time the application requests a connection it receives this pooled connection even if the preferred messaging engine has subsequently become available.

You can modify the connection pool settings to regularly discard all unused connections in the pool. After the connection pool is emptied, connections are made to the preferred messaging engine if one is available. For example, set the **ReapTime**, **AgedTimeout** and **UnusedTimeout** to 300 seconds, and the **PurgePolicy** to EntirePool. This refreshes the connection pool every 5 minutes, after which time the application selects a preferred messaging engine if one is available.

Required
It is required that a messaging engine is selected from the target group. A messaging engine in the target group is selected if one is available. If a messaging engine is not available in the target group, the connection process fails.

-targetTransportChain
The name of the inbound transport chain that the application should target when connecting to a messaging engine in a separate process to the application. If a messaging engine in another process is chosen, a connection can be made only if the messaging engine is in a server that runs the specified inbound transport chain. Refer to the information center for more information.

These transport chains specify the communication protocols that can be used to communicate with the application server to which the client application is connected.

If the selected messaging engine is in the same server as the application, a direct in-process connection is made and this transport chain property is ignored.

The transport chains represent network protocol stacks operating within a server. The name you specify must be one of the transport chains available in the server that hosts the messaging engine, as listed on the **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports** panel. The following transport chains are provided, but you can define your own transport chains on that panel.

InboundBasicMessaging
This is a connection-oriented protocol that uses a standard TCP/IP connection (JFAP-TCP/IP). It includes support for two-phase transactional (remote XA) flows, so that a message producer or consumer, running on a client or server system, can participate in a global transaction managed on that client or server system. The specific use for the XA flows is to support access from an application running in one server to a messaging engine on second server, perhaps because the first server does not have a suitable messaging engine. If the remote XA flows are used, a transaction coordinator must be available local to the application.

InboundSecureMessaging
This is the InboundBasicMessaging protocol wrapped in SSL.

For more information about using this property with other connection factory properties for workload management of connections, see the topic Administrative properties for JMS connections to a bus.

-providerEndPoints
A comma-separated list of endpoint triplets, with the syntax hostName:portNumber:chainName, used to connect to a bootstrap server. For example Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging. If hostName is not specified, the default is localhost. If portNumber is not specified, the default is 7276. If chainName is not specified, the default is BootstrapBasicMessaging. Refer to the information center for more information.

-connectionProximity
Bus | Host | Cluster | Server

-durableSubscriptionHome
me_name

-nonPersistentMapping
BestEffortNonPersistent | ExpressNonPersistent | ReliableNonPersistent | ReliablePersistent | AssuredPersistent | AsSIBDestination | None

-persistentMapping
BestEffortNonPersistent | ExpressNonPersistent | ReliableNonPersistent | ReliablePersistent | AssuredPersistent | AsSIBDestination | None

-readAhead
Default | AlwaysOn | AlwaysOff

-tempQueueNamePrefix
prefix

-tempTopicNamePrefix
prefix

-shareDurableSubscriptions
AsCluster | AlwaysShared | NeverShared

The “pass message payload by reference” properties:

-producerDoesNotModifyPayloadAfterSet
true | false (default false)

Applications that use this connection factory to send messages must obey the following rules:

- The application does not modify the data object contained in a JMS object message.
- The application populates a JMS bytes message by using a single call to writeBytes(byte[]) and does not modify the byte array after it is set in the message.

-consumerDoesNotModifyPayloadAfterGet
true | false (default false)

Applications that use this connection factory to receive messages must obey the following rule: The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION: The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

-authDataAlias
alias_name

-shareDataSourceWithCMP
True | False

-xaRecoveryAuthAlias
alias_name

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.createSIBJMSConnectionFactory("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)", ["-name", "jmscf1",
 "-jndiName", "jms/jmscf1", "-busName", "abus"])
'jmscf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733325084)'

wsadmin>AdminTask.createSIBJMSConnectionFactory("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml)", ["-name", "jmsqcf2",
 "-jndiName", "jms/jmsqcf1", "-busName", "abus", "-type", "queue"])
'jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733675578)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createSIBJMSConnectionFactory
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
 {-name jmscf1 -jndiName jms/jmscf1 -busName abus}
jmscf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733325084)

wsadmin>$AdminTask createSIBJMSConnectionFactory
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
 {-name jmsqcf2 -jndiName jms/jmsqcf1 -busName abus -type queue}
jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733675578)
```

deleteSIBJMSConnectionFactory command

Use the deleteSIBJMSConnectionFactory command to delete a JMS connection factory for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('SIBJMSAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('`*command_name*`')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes the specified JMS connection factory.

Target object

A JMS connection factory.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.deleteSIBJMSConnectionFactory("jmsqcf2(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#J2CConnectionFactory_1098736176544)")
'jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098736176544)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask deleteSIBJMSConnectionFactory
jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098736176544)
jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098736176544)
```

listSIBJMSConnectionFactories command

Use the listSIBJMSConnectionFactories command to list all JMS connection factories for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all JMS connection factories for the default messaging provider at a specific scope

Target object

Scope of the default messaging provider at which the JMS connection factories were created.

Required parameters

None.

Optional parameters

-type

The **-type** parameter is used to filter the list of connection factories.

Select one of the following values:

all List all JMS connection factories (generic, queue, and topic) at the specified scope.

queue List all JMS queue connection factories at the specified scope.

topic List all JMS topic connection factories at the specified scope.

If the **-type** parameter is not supplied, then only generic JMS connection factories at the scope are listed.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

AdminTask.listSIBJMSConnectionFactories("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
'qcf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098730054140)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listSIBJMSConnectionFactories
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
qcf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098730054140)
```

modifySIBJMSConnectionFactory command

Use the modifySIBJMSConnectionFactory command to modify the properties of a JMS connection factory for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command changes the properties of a JMS connection factory.

Target object

A JMS connection factory.

Required parameters

None.

Optional parameters

-name
factory_name

-jndiName
jndi_name

-category
category

-description
text

-logMissingTransactionContext
True | False

-manageCachedHandles
True | False

-busName
name

-clientID
id

-userName
name

-password
password

-target
target_name

-targetType
BusMember | Custom | ME

-targetSignificance
Preferred | Required

-targetTransportChain
transport_chain

-providerEndPoints
tuple_list

-connectionProximity
Bus | Host | Cluster | Server

-durableSubscriptionHome
me_name

-nonPersistentMapping
BestEffortNonPersistent | ExpressNonPersistent | ReliableNonPersistent | ReliablePersistent | AssuredPersistent | AsSIBDestination | None

-persistentMapping
BestEffortNonPersistent | ExpressNonPersistent | ReliableNonPersistent | ReliablePersistent | AssuredPersistent | AsSIBDestination | None

-readAhead
Default | AlwaysOn | AlwaysOff

-tempQueueNamePrefix
prefix

-tempTopicNamePrefix
prefix

-shareDurableSubscriptions
AsCluster | AlwaysShared | NeverShared

The “pass message payload by reference” properties:

-producerDoesNotModifyPayloadAfterSet
true | false (default false)

Applications that use this connection factory to send messages must obey the following rules:

- The application does not modify the data object contained in a JMS object message.
- The application populates a JMS bytes message by using a single call to writeBytes(byte[]) and does not modify the byte array after it is set in the message.

-consumerDoesNotModifyPayloadAfterGet
true | false (default false)

Applications that use this connection factory to receive messages must obey the following rule: The application does not modify the data object obtained from a JMS object message. The data object is treated as read only.

When large object messages or bytes messages are sent, the cost in memory and processor use of serializing, deserializing, and copying the message payload can be significant. If you enable the **pass message payload by reference** properties on a connection factory or activation specification, you tell the default messaging provider to override the JMS 1.1 specification and potentially reduce or bypass this data copying.

CAUTION: The parts of the JMS Specification that are bypassed by these properties are defined to ensure message data integrity. Any of your JMS applications that use these properties must strictly follow the rules that are described in the topic Why and when to pass the JMS message payload by reference, or you risk losing data integrity.

-authDataAlias
alias_name

-shareDataSourceWithCMP
True | False

-xaRecoveryAuthAlias
alias_name

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.modifySIBJMSConnectionFactory("jmsqcf2(cells/9994GKCNode01Cell/
```

```
nodes/9994GKCNode01|resources.xml#J2CConnectionFactory_1098736176544)",
 ["-manageCachedHandles", "True"])
'jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733675578)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask modifySIBJMSConnectionFactory
jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098736176544)
 {-manageCachedHandles True}
jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733675578)
```

showSIBJMSConnectionFactory command

Use the showSIBJMSConnectionFactory command to show the properties of a JMS connection factory for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command returns a set of property-value pairs for the specified JMS connection factory.

Target object

A JMS connection factory.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.showSIBJMSConnectionFactory("jmsqcf2(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#J2CConnectionFactory_1098733675578)")
'{target=, targetTransportChain=, readAhead=Default, password=,
targetType=BusMember, tempQueueNamePrefix=, connectionProximity=Bus,
```

```
nonPersistentMapping=ExpressNonPersistent, name=jmsqcf2,
targetSignificance=Preferred, shareDurableSubscriptions=InCluster,
providerEndPoints=, shareDataSourceWithCMP=false, userName=,
logMissingTransactionContext=false, busName=abus,
persistentMapping=ReliablePersistent, clientID=,
jndiName=jms/jmsqcf1, manageCachedHandles=false}'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask showSIBJMSConnectionFactory
jmsqcf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CConnectionFactory_1098733675578)
{target=, targetTransportChain=, readAhead=Default, password=,
targetType=BusMember, tempQueueNamePrefix=, connectionProximity=Bus,
nonPersistentMapping=ExpressNonPersistent, name=jmsqcf2,
targetSignificance=Preferred, shareDurableSubscriptions=InCluster,
providerEndPoints=, shareDataSourceWithCMP=false, userName=,
logMissingTransactionContext=false, busName=abus,
persistentMapping=ReliablePersistent, clientID=,
jndiName=jms/jmsqcf1, manageCachedHandles=false}
```

createSIBJMSQueue command

Use the createSIBJMSQueue command to create a new JMS queue for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The createSIBJMSQueue command creates a JMS queue for the default messaging provider at a specific scope.

Target object

The scope of the default messaging provider at which the JMS queue is to be created.

Required parameters

-name
The identifier by which this JMS queue is known for administrative purposes.

-jndiName
The JNDI name that is used to bind the queue into the application server namespace.

-queueName
The name of the service integration bus destination to which the JMS queue maps.

Optional parameters

-description
An optional description for the bus, for administrative purposes.

-deliveryMode Application | NonPersistent | Persistent
The delivery mode to be used by MessageProducers for messages sent to this queue.

-timeToLive
The default length of time from its dispatch time that a message sent to this queue should be retained by the system, where 0 indicates that time to live value does not expire. The value from the producer is used if the **Time to Live** parameter is not supplied.

-priority
The priority for messages sent to this queue. The value from the producer is used if not completed. In the range 0 to 9 where 0 is the lowest priority and 9 is the highest priority

-readAhead AsConnection | AlwaysOn | AlwaysOff
Used to control read-ahead optimization during message delivery. The default is AsConnection.

-busName
The name of the service integration bus that the service integration bus destination, identified by **queueName**, is configured on. If not set, the bus that the application is connected to is used.

-scopeToLocalQP TRUE | FALSE
Indicates whether the underlying service integration bus queue destination is scoped to a local queue point when addressed using this JMS queue. A local queue point is a queue point that is configured on the messaging engine to which the JMS application is connected. The option applies when using this JMS queue to send and receive messages and when setting a reply queue in a request message. When a reply queue is set in a request message, the local queue point is on the messaging engine to which the application setting the reply queue is connected, not the messaging engine to which the application that uses the reply queue sends the reply message. If the connected messaging engine does not have a queue point for the destination this option is ignored. The default value is FALSE.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

-producerBind TRUE | FALSE
Indicates how JMS producers bind to queue points of the clustered queue. The default value is FALSE.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

TRUE The messaging system selects a queue point when the session is opened. All messages produced by the session are sent to the chosen queue point. The messaging system uses the **producerPreferLocal** setting when selecting the queue point.

FALSE
The messaging system selects a queue point each time a message is sent, potentially

workload balancing the messages across all available queue points. The messaging system uses the **producerPreferLocal** setting when selecting the queue point.

-producerPreferLocal TRUE | FALSE
Indicates whether a queue point local to the producer is preferred to other available queue points when the messaging system selects a queue point to produce messages to. A local queue point is a queue point that is configured on the messaging engine to which the JMS application is connected. The default value is TRUE.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

-consumerGatherMessages TRUE | FALSE
A JMS consumer or browser is attached to a single queue point of the service integration bus destination by the messaging system. This parameter indicates whether a JMS consumer or browser take messages from any available queue points of the service integration bus destination (TRUE), or the single queue point to which it is attached (FALSE). The default value is FALSE. Gathering messages from multiple queue points results in an increased performance cost and message order cannot be maintained.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Examples

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.createSIBJMSQueue("WASINSTALL2Node01(cells/WASINSTALL2Cell01/
nodes/WASINSTALL2Node01|node.xml#Node_1)", ["-name", "jmsq2", "-jndiName",
 "jms/jnmsq2", "-queueName", "busq4jmsq2"])
'jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml
#J2CAdminObject_1098737234986)'

wsadmin>AdminTask.listSIBJMSQueues("WASINSTALL2Node01(cells/WASINSTALL2Cell01
/nodes/WASINSTALL2Node01|node.xml#Node_1)")
'queue1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098711838691)
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createSIBJMSQueue
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
 {-name jmsq2 -jndiName jms/jnmsq2 -queueName busq4jmsq2}
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)

wsadmin>$AdminTask listSIBJMSQueues
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
queue1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098711838691)
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)
```

The following example shows how to create a default messaging provider JMS queue that selects a queue point when a session is opened and never changes:

- Using Jython:

```
wsadmin>AdminTask.createSIBJMSQueue("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name", "MyJMSQueue", "-jndiName",
 "MyJMSQueue", "-busName bus1", "-queueName", "MyExistingQueue", "-deliveryMode",
 "Application", "-readAhead", "AsConnection", "-producerBind", "TRUE"])
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBJMSQueue 9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1 {-name MyJMSQueue -jndiName MyJMSQueue
-busName bus1 -queueName MyExistingQueue -deliveryMode Application
-readAhead AsConnection -producerBind TRUE }
```

The following example shows how to create a default messaging provider JMS queue that selects a queue point every time a message is sent. There is no preference over which queue point is selected and the consumers take messages from any queue point:

- Using Jython:

```
wsadmin>AdminTask.createSIBJMSQueue("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name", "MyJMSQueue", "-jndiName",
 "MyJMSQueue", "-busName bus1", "-queueName", "MyExistingQueue", "-deliveryMode",
 "Application", "-readAhead", "AsConnection", "-producerBind", "FALSE",
 "-producerPreferLocal", "FALSE", "-gatherMessages", "TRUE"])
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBJMSQueue 9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1 {-name MyJMSQueue -jndiName MyJMSQueue
-busName bus1 -queueName MyExistingQueue -deliveryMode Application
-readAhead AsConnection -producerBind FALSE -producerPreferLocal FALSE
-gatherMessages TRUE}
```

The following example shows how to create a default messaging provider JMS queue that scopes all operations on it down to the queue point local to the user of the JMS queue:

- Using Jython:

```
wsadmin>AdminTask.createSIBJMSQueue("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name", "MyJMSQueue", "-jndiName",
 "MyJMSQueue", "-busName bus1", "-queueName", "MyExistingQueue", "-deliveryMode",
 "Application", "-readAhead", "AsConnection", "-scopeToLocalQP", "TRUE"])
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBJMSQueue 9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1 {-name MyJMSQueue -jndiName MyJMSQueue
-busName bus1 -queueName MyExistingQueue -deliveryMode Application
-readAhead AsConnection -scopeToLocalQP TRUE}
```

deleteSIBJMSQueue command

Use the deleteSIBJMSQueue command to delete a JMS queue for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBJMSAdminCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes the specified JMS queue.

Target object

A JMS queue.

The JMS queue is deleted at the specified scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.deleteSIBJMSQueue("jmsq2(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CAdminObject_1098737234986)")
'jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask deleteSIBJMSQueue
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)
```

listSIBJMSQueues command

Use the listSIBJMSQueues command to list all JMS queues for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command returns a list of all JMS queues for the default messaging provider at a specific scope.

Target object

Scope of the default messaging provider at which the JMS queues were created.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

  ```
  wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
  '9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

  wsadmin>AdminTask.listSIBJMSQueues("9994GKCNode01(cells/9994GKCNode01Cell/
  nodes/9994GKCNode01|node.xml#Node_1)")
  'queue1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
  J2CAdminObject_1098711838691)'
  ```
- Using Jacl:

  ```
  wsadmin>$AdminConfig getid /Node:9994GKCNode01
  9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

  wsadmin>$AdminTask listSIBJMSQueues
  9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
  queue1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
  J2CAdminObject_1098711838691)
  ```

modifySIBJMSQueue command

Use the modifySIBJMSQueue command to change the properties of a JMS queue for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The modifySIBJMSQueue command changes the properties of a JMS queue for the default messaging provider at a specific scope.

Target object

A JMS queue.

Required parameters

None.

Optional parameters

-name
The identifier by which this JMS queue is known for administrative purposes.

-jndiName
The JNDI name that is used to bind the queue into the application server namespace.

-queueName
The name of the service integration bus destination to which the JMS queue maps.

-description
An optional description for the bus, for administrative purposes.

-deliveryMode Application | NonPersistent | Persistent
The delivery mode to be used by MessageProducers for messages sent to this queue.

-timeToLive
The default length of time from its dispatch time that a message sent to this queue should be retained by the system, where 0 indicates that time to live value does not expire. Value from the producer is used if this parameter is not supplied.

-priority
The priority for messages sent to this queue. The value from the producer is used if not completed. In the range 0 to 9 where 0 is the lowest priority and 9 is the highest priority

-readAhead AsConnection | AlwaysOn | AlwaysOff
Used to control read-ahead optimization during message delivery. The default is AsConnection.

-busName
The name of the service integration bus that the bus destination, identified by **queueName**, is configured on. If not set, the bus that the application is connected to is used.

-scopeToLocalQP TRUE | FALSE
Indicates whether the underlying service integration bus queue destination is scoped to a local queue point when addressed using this JMS queue. A local queue point is a queue point that is configured on the messaging engine to which the JMS application is connected. The option applies when using this JMS queue to send and receive messages and when setting a reply queue in a request message. When a reply queue is set in a request message, the local queue point is on the messaging engine to which the application setting the reply queue is connected, not the messaging engine to which the

application using the reply queue sends the reply message. If the connected messaging engine does not have a queue point for the destination this option is ignored. The default value is FALSE.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server results in an exception to the application.

-producerBind TRUE | FALSE
Indicates how JMS producers bind to queue points of the clustered queue. The default value is FALSE.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server results in an exception to the application.

TRUE The messaging system selects a queue point when the session is opened. All messages produced by the session are sent to the chosen queue point. The messaging system uses the **producerPreferLocal** setting when selecting the queue point.

FALSE
The messaging system selects a queue point each time a message is sent, potentially workload balancing the messages across all available queue points. The messaging system uses the **producerPreferLocal** setting when selecting the queue point.

-producerPreferLocal TRUE | FALSE
Indicates whether a queue point local to the producer is preferred to other available queue points when the messaging system selects a queue point to produce messages to. A local queue point is a queue point that is configured on the messaging engine to which the JMS application is connected. The default value is TRUE.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server results in an exception to the application.

-consumerGatherMessages TRUE | FALSE
A JMS consumer or browser is attached to a single queue point of the service integration bus destination by the messaging system. This parameter indicates whether a JMS consumer or browser take messages from any available queue points of the service integration bus destination (TRUE), or the single queue point to which it is attached (FALSE). The default value is FALSE. Gathering messages from multiple queue points results in an increased performance cost and message order cannot be maintained.

Changing the default setting of this option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on a WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server results in an exception to the application.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.modifySIBJMSQueue("jmsq2(cells/9994GKCNode01Cell/nodes/
```

```
9994GKCNode01|resources.xml#J2CAdminObject_1098737234986)",
["-queueName", "q2forjms"])
'jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask modifySIBJMSQueue
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)
 {-queueName q2forjms}
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)
```

showSIBJMSQueue command

Use the showSIBJMSQueue command to show properties of a JMS queue for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command returns a set of property-value pairs for the specified JMS queue.

Target object

A JMS queue.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.showSIBJMSQueue("jmsq2(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CAdminObject_1098737234986)")
'{jndiName=jms/jnmsq2, deliveryMode=Application, busName=, name=jmsq2,
readAhead=AsConnection, timeToLive=, priority=, queueName=busq4jmsq2}'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask showSIBJMSQueue
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098737234986)
{jndiName=jms/jnmsq2, deliveryMode=Application, busName=, name=jmsq2,
readAhead=AsConnection, timeToLive=, priority=, queueName=busq4jmsq2}
```

createSIBJMSTopic command

Use the createSIBJMSTopic command to create a new JMS topic for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a JMS topic for the default messaging provider at a specific scope.

Target object

Scope of the default messaging provider at which the JMS topic is to be created.

Required parameters

```
-name jmstopic_name
-jndiName jndi_name
```

Optional parameters

```
-description text
-topicName topic_name
-topicSpace topicspace_name
-deliveryMode Application | NonPersistent | Persistent
-timeToLive time
-priority priority
-readAhead AsConnection | AlwaysOn | AlwaysOff
-busName name
```

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.createSIBJMSTopic("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name", "jmstopic2", "-jndiName",
 "jms/jnmstopic2", "-topicSpace", "sportshall"])
'jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)'

wsadmin>AdminTask.listSIBJMSTopics("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
'topic1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738449292)
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createSIBJMSTopic
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
 {-name jmstopic2 -jndiName jms/jnmstopic2 -topicSpace sportshall}
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)

wsadmin>$AdminTask listSIBJMSTopics
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
topic1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738449292)
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)
```

deleteSIBJMSTopic command

Use the deleteSIBJMSTopics command to delete a JMS topic for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('SIBJMSAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('`*command_name*`')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes the specified JMS topic.

Target object

A JMS topic.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.deleteSIBJMSTopic("jmstopic2(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CAdminObject_1098738992263)")
'jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask deleteSIBJMSTopic
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)
```

listSIBJMSTopics command

Use the listSIBJMSTopics command to list all JMS topics for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command returns a list of all JMS topics for the default messaging provider at a specific scope.

Target object

Scope of the default messaging provider at which the JMS topics were created.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.listSIBJMSTopics("9994GKCNode01(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|node.xml#Node_1)")
'topic1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738449292)'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listSIBJMSTopics
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
topic1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738449292)
```

modifySIBJMSTopic command

Use the modifySIBJMSTopic command to change the properties of a JMS topic for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command changes the properties of a JMS topic for the default messaging provider at a specific scope.

Target object

A JMS topic.

Required parameters

None.

Optional parameters

```
-name jmstopic_name
-jndiName jndi_name
-description text
-topicName topic_name
-topicSpace topicspace_name
-deliveryMode Application | NonPersistent | Persistent
-timeToLive time
-priority priority
-readAhead AsConnection | AlwaysOn | AlwaysOff
-busName name
```

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
'9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

wsadmin>AdminTask.modifySIBJMSTopic("jmstopic2(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CAdminObject_1098738992263)", ["-jndiName",
 "jms/jnmstopic2", "-topicName", "archery", "-readAhead", "AlwaysOn"])
'jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)'

wsadmin>AdminTask.showSIBJMSTopic("jmstopic2(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CAdminObject_1098738992263)")
'{topicSpace=sportshall, deliveryMode=Application, jndiName=jms/jnmstopic2,
busName=, readAhead=AlwaysOn, name=jmstopic2, timeToLive=, priority=,
topicName=archery}'
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask modifySIBJMSTopic
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)
 {-jndiName jms/jnmstopic2 -topicName archery -readAhead AlwaysOn}
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)

wsadmin>$AdminTask showSIBJMSTopic
jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CAdminObject_1098738992263)
{topicSpace=sportshall, deliveryMode=Application, jndiName=jms/jnmstopic2,
busName=, readAhead=AlwaysOn, name=jmstopic2, timeToLive=, priority=,
topicName=archery}
```

showSIBJMSTopic command

Use the showSIBJMSTopic command to show properties of a JMS topic for the default messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus JMS commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBJMSAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command returns a set of property-value pairs for the specified JMS topic.

Target object

A JMS topic.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

  ```
  wsadmin>AdminConfig.getid("/Node:9994GKCNode01" )
  '9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)'

  wsadmin>AdminTask.showSIBJMSTopic("jmstopic2(cells/9994GKCNode01Cell/nodes/
  9994GKCNode01|resources.xml#J2CAdminObject_1098738992263)")
  '{topicSpace=sportshall, deliveryMode=Application, jndiName=jms/jnmstopic2,
  busName=, readAhead=AsConnection, name=jmstopic2, timeToLive=, priority=,
  topicName=jmstopic2}'
  ```

- Using Jacl:

  ```
  wsadmin>$AdminConfig getid /Node:9994GKCNode01
  9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

  wsadmin>$AdminTask showSIBJMSTopic
  jmstopic2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
  J2CAdminObject_1098738992263)
  {topicSpace=sportshall, deliveryMode=Application, jndiName=jms/jnmstopic2,
  busName=, readAhead=AsConnection, name=jmstopic2, timeToLive=, priority=,
  topicName=jmstopic2}
  ```

Chapter 13. Managing messaging with the WebSphere MQ messaging provider

Through the WebSphere MQ messaging provider in WebSphere Application Server, Java Message Service (JMS) messaging applications can use your WebSphere MQ system as an external provider of JMS messaging resources. To enable this approach, you configure the WebSphere MQ messaging provider in WebSphere Application Server to define JMS resources for connecting to any queue manager on the WebSphere MQ network.

Before you begin

If your business uses WebSphere MQ, and you want to integrate WebSphere Application Server messaging applications into a predominantly WebSphere MQ network, the WebSphere MQ messaging provider is a logical choice. However, there can be benefits in using another provider. If you are not sure which provider combination is best suited to your requirements, see Choosing messaging providers for a mixed environment.

The preferred solution for publish and subscribe messaging with WebSphere MQ as an external JMS messaging provider is to use a message broker such as WebSphere MQ Event Broker.

About this task

The WebSphere MQ messaging provider supports JMS 1.1 domain-independent interfaces (sometimes referred to as “unified” or “common” interfaces), and also supports the Java EE Connector Architecture (JCA) 1.5 activation specification mechanism for message-driven beans (MDBs) across all platforms supported by WebSphere Application Server.

You can use WebSphere Application Server to configure WebSphere MQ resources for applications (for example queue connection factories) and to manage messages and subscriptions associated with JMS destinations. You administer security through WebSphere MQ.

You can use WebSphere Application Server to coordinate global transactions including WebSphere MQ without configuring the extended transactional client.

For publish and subscribe messaging with WebSphere MQ as an external JMS messaging provider you have several options:

- With WebSphere MQ Version 7 on any platform you can use the built-in publish and subscribe capability of WebSphere MQ. Note that you cannot use WebSphere MQ Message Broker Version 7 for this because it no longer provides a publish and subscribe capability.

Procedure

- Learn about the WebSphere MQ messaging provider.
- Configure JMS resources for the WebSphere MQ messaging provider.

  You can do this through the WebSphere Application Server administrative console, or through the WebSphere Application Server set of WebSphere MQ administrative commands.
- List JMS resources for the WebSphere MQ messaging provider.

JMS provider settings

Use this panel to view the configuration properties of a selected JMS provider.

To view this page, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->JMS providers**. This displays a list of JMS providers in the content pane. For each JMS provider in the list, the entry indicates the *scope* level at which JMS resource definitions are visible to applications. You can create the same type of JMS provider at different Scope settings, to offer JMS resources at different levels of visibility to applications.
2. (optional) If you want to manage JMS resources that are defined at a different scope setting, change the **Scope** setting to the required level.
3. In the Providers column of the list displayed, click the name of a JMS provider.

If you want to browse or change JMS resources of the JMS provider, click the link for the type of resource under Additional Properties. For more information about the administrative console panels for the types of JMS resources, see the related topics.

For the default messaging provider (which is based on service integration technologies) and the WebSphere MQ messaging provider, the scope, name, and description properties are displayed for information only. You cannot change these properties.

For a third-party non-JCA provider that you have defined yourself, the properties of that provider are displayed.

Scope

The level (cell, node, or server level) at which this resource definition is available.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes. For more information about the scope setting, see Scope settings.

Name

The name by which the JMS provider is known for administrative purposes.

Data type

String

Default

- Default messaging provider.

  For JMS resources to be provided by a service integration bus, as part of WebSphere Application Server.
- *My JMSprovider*

  For JMS resources to be provided by a third-party JMS provider that you specify, rather than by the default messaging provider or the WebSphere MQ messaging provider that are available as part of WebSphere Application Server. You assign the name, for example “My JMSprovider”, when you define the third-party JMS provider to WebSphere Application Server. You must also have installed and configured the third-party JMS provider before applications can use its JMS resources.
- WebSphere MQ messaging provider

  For JMS resources to be provided by WebSphere MQ. You must have installed and configured WebSphere MQ before applications can use its JMS resources.
- V5 default messaging provider.

  For JMS resources to be provided by a WebSphere Application Server Version 5 node.

Description

A description of the JMS provider, for administrative purposes within WebSphere Application Server.

Data type String

Classpath

A list of paths or JAR file names that together form the location for the JMS provider classes. Each class path entry is on a separate line (separated by using the Enter key) and must not contain path separator characters (such as ';' or ': '). Class paths can contain variable (symbolic) names to be substituted by using a variable map. Check your driver installation notes for specific JAR file names that are required.

Note: This property is only available for third-party messaging providers.

Data type String

Native library path

An optional path to any native libraries (*.dll, *.so). Each native path entry is on a separate line (separated by using the Enter key) and must not contain path separator characters (such as ';' or ': '). Native paths can contain variable (symbolic) names to be substituted by using a variable map.

Note: This property is only available for the WebSphere MQ messaging provider and third-party messaging providers.

Data type String

Update resource adapter

This button can be used to update the WebSphere MQ resource adapter that provides the function made available by the WebSphere MQ messaging provider. This button must only be used as directed by a member of IBM service, otherwise it may result in the use of an unsupported level of the WebSphere MQ resource adapter.

Normally the WebSphere MQ resource adapter is automatically updated by applying WebSphere Application Server fix packs. It is important to note that use of the **Update resource adapter** button prevents these automatic updates from happening for future fix packs for any node on which the button is used. If, in the future, you require the WebSphere MQ resource adapter used by the node to receive updates when a fix pack is applied then you must re-establish the recommended resource adapter configuration. For more information see “Maintaining the WebSphere MQ resource adapter” on page 724.

Note: This property is only available for the WebSphere MQ messaging provider.

Data type Button

External initial context factory

The Java classname of the initial context factory for the JMS provider.

For example, for an LDAP service provider the value has the form: com.sun.jndi.ldap.LdapCtxFactory.

Note: This property is only available for third-party messaging providers.

Data type String
Default Null

External provider URL

The JMS provider URL for external JNDI lookups.

For example, an LDAP URL for a messaging provider has the form: `ldap://hostname.company.com/contextName`.

Note: This property is only available for third-party messaging providers.

Data type	String
Default	Null

Disable WebSphere MQ

This check box is only valid for the WebSphere MQ messaging provider. When selected, this check box disables all WebSphere MQ functionality on affected application servers. Note that you must restart the affected application server processes for this change to take effect.

In a single server environment this check box is only available on the WebSphere MQ messaging provider panel where the scope is set to server, and has the effect of disabling all WebSphere MQ functionality in that application server.

In a network deployment environment this check box is available on all WebSphere MQ messaging provider panels. The effect of selecting the check box depends on the scope at which you select it:

- At the cell scope, all WebSphere MQ functionality is disabled on all application servers in the cell.
- At the node scope, all WebSphere MQ functionality is disabled on all application servers that are part of that node.
- At the cluster scope, all WebSphere MQ functionality is disabled on all application servers in that cluster.
- At the server scope, all WebSphere MQ functionality is disabled in that particular application server.

The value of the check box at a higher scope takes precedence over the value at a lower scope. For example, if you do not select the check box for a WebSphere MQ messaging provider at the server scope, but do select it for a WebSphere MQ messaging provider at a higher scope (such as the cell scope), the value of the check box at the cell scope takes precedence and WebSphere MQ functionality is therefore disabled in all application servers in the cell, regardless of whether the check box is selected at the server scope.

An informational message indicating that WebSphere MQ has been disabled is added to all WebSphere MQ messaging provider panels that are at affected scopes, but this message does not appear on those panels where the check box is selected. In a single server environment this informational message is only displayed after a server restart is performed. In a network deployment environment the informational message is displayed immediately.

For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Data type	Check box
Default	Not selected

Additional properties

Connection factories

A connection factory is used to create connections to the associated JMS provider. These connection factories can be used for accessing JMS Queue and JMS Topic destinations.

Queue connection factories
A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for point-to-point messaging.

Topic connection factories
A topic connection factory is used to create connections to the associated JMS provider of JMS topic destinations, for publish and subscribe messaging.

Queues
A JMS queue is used as a destination for point-to-point messaging.

Topics
A JMS topic is used as a destination for publish/subscribe messaging.

Activation specifications
A JMS activation specification is associated with one or more message-driven beans and provides configuration necessary for them to receive messages.

Resource adapter properties
These properties are used to configure the WebSphere MQ resource adapter, which is used by the WebSphere MQ messaging provider. In particular most of these settings affect the behavior of WebSphere MQ messaging provider activation specifications.

Resource adapter properties

Use this page to configure the WebSphere MQ resource adapter that underlies the WebSphere MQ messaging provider.

To view this page, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->JMS providers** to display a list of JMS providers in the content pane.
2. (optional) If you want to manage JMS resources that are defined at a different scope setting, change the **Scope** setting to the required level.
3. In the Providers column of the displayed list of JMS providers, click the name of the WebSphere MQ messaging provider that you want to work with.
4. In the content pane under Additional properties, click **Resource adapter properties** to view the configuration page for the properties.

The resource adapter properties are used to configure the WebSphere MQ resource adapter, which is used by the WebSphere MQ messaging provider. These properties affect the connection pool, which is used by the WebSphere MQ messaging provider activation specifications. They do not affect the WebSphere MQ messaging provider queues, topics, or connection factories.

These properties only have an effect on the WebSphere MQ messaging provider objects that are defined at the same scope as the messaging provider and resource adapter on which they are set. So, for example, if you set the max connections property to a particular setting at the server scope, only the server scoped WebSphere MQ messaging provider activation specifications are affected by this setting.

For further information about these properties, see Configuration of the ResourceAdapter object in the WebSphere MQ information center.

If you want to configure a WebSphere MQ resource adapter custom property that is not exposed in WebSphere Application Server, click **Custom properties** under Additional properties.

Connection pool properties

Max connections

The maximum number of connections to a WebSphere MQ queue manager.

Data type	String
Default	10

Connection concurrency

The maximum number of message-driven beans that can be supplied by each connection.

Data type	String
Default	5

Reconnection retry count

The maximum number of attempts made by a WebSphere MQ messaging provider activation specification to reconnect to a WebSphere MQ queue manager if a connection fails.

Data type	String
Default	5

Reconnection retry interval

The time, in milliseconds, that a WebSphere MQ messaging provider activation specification waits before making another attempt to reconnect to a WebSphere MQ queue manager.

Data type	String
Default	300000

Additional properties

Custom properties

The full set of custom properties that are used to configure the WebSphere MQ resource adapter. Use this page to configure custom properties that are not exposed in WebSphere Application Server.

Installing WebSphere MQ to interoperate with WebSphere Application Server

When you install a new WebSphere MQ network, you can tune the installation for working with WebSphere Application Server. If you have an established WebSphere MQ network, you can choose whether to modify some of the settings for better interoperation.

About this task

This topic provides installation instructions for setting up a new WebSphere MQ installation to interoperate with WebSphere Application Server. If you have an established WebSphere MQ network, treat this task as a source of tips to tune your existing WebSphere MQ installation.

Procedure

1. Install a supported version of WebSphere MQ, as described in the installation instructions provided with WebSphere MQ.

To identify the supported version of WebSphere MQ, see the following article: Detailed system requirements page.

You should not install Rational Application Developer and WebSphere Application Server on the same machine when using WebSphere MQ.

For other installation prerequisites, see the *Quick Beginnings* section for your platform, in the WebSphere MQ information center.

2. Follow the WebSphere MQ instructions for verifying your installation setup.
3. Configure WebSphere Application Server and WebSphere MQ to interoperate effectively.

   For information about the prerequisites and requirements for effective interoperation, see the following technote: Information about using the WebSphere MQ messaging provider for WebSphere Application Server Version 7.0.
4. Configure the WebSphere MQ messaging provider with native libraries information.

   To connect to a WebSphere MQ queue manager or queue-sharing group in bindings mode, the WebSphere MQ messaging provider needs to know where to load native libraries from. For more information, see "Configuring the WebSphere MQ messaging provider with native libraries information."

   Note: For migration purposes only, you can also specify native path information, when in an application server environment, by setting the *MQ_INSTALL_ROOT* WebSphere Application Server environment variable. For more information see, Installing WebSphere MQ to interoperate with WebSphere Application Server.
5. Optional: At Cell scope or Node scope, set the WebSphere Application Server MQ_CLEAR_MQ_FROM_OSGI_CACHE_ON_SHUTDOWN environment variable to `True`. This allows application server startup to automatically take account of changes that are made to the MQ_INSTALL_ROOT environment variable and WebSphere MQ JMS client libraries while the application server is stopped.

   If you do not set this variable, you must restart the application server a second time after any changes of this type, to enable the application to perform messaging by using the WebSphere MQ messaging provider.

   Note: If you set the MQ_CLEAR_MQ_FROM_OSGI_CACHE_ON_SHUTDOWN environment variable, the startup time might increase because, on startup, each application server needs to initialize an additional state associated with WebSphere MQ installation.

What to do next

You are now ready to configure a messaging provider. If your business uses WebSphere MQ, and you want to integrate WebSphere Application Server messaging applications into a predominantly WebSphere MQ network, the WebSphere MQ messaging provider is the natural choice. However, there can be benefits in using another provider. If you are not sure which provider combination is best suited to your needs, see Choosing messaging providers for a mixed environment.

To create WebSphere MQ messaging provider resources, see "Configuring JMS resources for the WebSphere MQ messaging provider" on page 734.

Configuring the WebSphere MQ messaging provider with native libraries information

To connect to a WebSphere MQ queue manager or queue-sharing group in bindings mode, the WebSphere MQ messaging provider needs to know where to load native libraries from. This information is known as native path information. The way native path information is set depends on whether the connection is established in an application client or in an application server environment.

About this task

If you are running in a client environment, use launchClient to start a client application. In the system property `MQ_INSTALL_ROOT` enter the name of a directory that contains the WebSphere MQ native libraries, in a subdirectory of java/lib or java/lib64 depending on whether you are using 32 bit or 64 bit native libraries. For example, on Linux specify `./launchClient.sh myappclient.ear -CCDMQ_INSTALL_ROOT=/opt/mqm/`.

If you are running in an application server environment, you can configure the WebSphere MQ messaging provider with native path information by using the command line, as described in "WMQAdminCommands command group for the AdminTask object" on page 870, or you can use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, expand **Resources** > **JMS** > **JMS providers**.
2. Select the WebSphere MQ messaging provider that is at the correct **Scope** for the connection factory or activation specification that will create the bindings mode connection.

   Note:

   - Native path information at **Server** scope is used in preference to native path information at all other scopes.
   - Native path information at **Cluster** scope is used in preference to native path information at **Node** and **Cell** scopes.
   - Native path information at **Node** scope is used in preference to native path information at **Cell** scope.
3. Under General Properties, in the **Native library path** property, enter the full name of the directory that contains the WebSphere MQ native libraries. For example, on Linux enter `/opt/mqm/java/lib`. Enter only one directory name.
4. Click **OK**.
5. Save any changes to the master configuration.
6. If you are running in an application server environment, you must restart all affected servers twice when you have changed the native path information. Otherwise, a WMSG1623E message is produced, indicating that the WebSphere MQ messaging provider is not available.

   If you are running in a client environment, you must rerun the client program twice. Otherwise, a WMSG2013E message is produced.

   Whichever environment you are running in, until you perform these restarts any attempt to use a WebSphere MQ messaging provider resource (for example, a connection factory) from one of the affected servers causes a javax.naming.NamingException and a WMSG2003E message.

What to do next

Note: For migration purposes only, you can also specify native path information, when in an application server environment, by setting the *MQ_INSTALL_ROOT* WebSphere Application Server environment variable. For more information, see the following topic in the WebSphere Application Server Version 6.1 information center: Installing WebSphere MQ to interoperate with WebSphere Application Server (Version 6.1).

Maintaining the WebSphere MQ resource adapter

The WebSphere MQ resource adapter is used by all applications that perform JMS messaging with the WebSphere MQ messaging provider. The WebSphere MQ resource adapter is usually updated automatically when you apply WebSphere Application Server fix packs, but if you have previously manually updated the resource adapter you must manually update your configuration to ensure that maintenance is applied correctly.

About this task

Applying a WebSphere Application Server fix pack does not automatically update the version of the WebSphere MQ resource adapter used by servers on nodes where the WebSphere MQ resource adapter has previously been manually updated.

Procedure

- To migrate the configuration of all servers in the cell to use the latest version of the WebSphere MQ resource adapter contained in the WebSphere Application Server installation, see “Ensuring that servers use the latest available WebSphere MQ resource adapter maintenance level.”
- If you want a specific version of the WebSphere MQ resource adapter to be installed, and the version you require is not available in a WebSphere Application Server fix pack, or an interim fix, see “Installing a specific maintenance level of the WebSphere MQ resource adapter” on page 726.

Ensuring that servers use the latest available WebSphere MQ resource adapter maintenance level

To ensure that the WebSphere MQ resource adapter is automatically updated to the latest available maintenance level when you apply WebSphere Application Server fix packs, you can configure all servers in your environment to use the latest version of the resource adapter contained in the WebSphere Application Server fix pack that you have applied to the installation of each node.

About this task

Use this task if any of the following circumstances apply to your configuration, and you want to configure all servers in your environment to use the latest version of the WebSphere MQ resource adapter:

- The JVM logs of any application server in your environment contain the following entry:

  ```
  WMSG1625E: It was not possible to detect
  the WebSphere MQ messaging provider code at the specified path <null>
  ```

- One or more nodes has previously been manually updated to use a specific maintenance level of the WebSphere MQ resource adapter that is now superseded by the latest version of the resource adapter contained in the current WebSphere Application Server maintenance level.

When you have performed the following steps for all cells and single server installations in your environment, your servers will automatically receive maintenance to the WebSphere MQ resource adapter when a new WebSphere Application Server fix pack is applied.

Procedure

1. Start the application server.
2. Copy the following Jython script into a file called `convertWMQRA.py`, then save it into the *profile_root*`/bin` directory.

   ```
   ras = AdminUtilities.convertToList(AdminConfig.list('J2CResourceAdapter'))

   for ra in ras :
      desc = AdminConfig.showAttribute(ra, "description")
      if (desc == "WAS 7.0 Built In WebSphere MQ Resource Adapter") or
         (desc == "WAS 7.0.0.1 Built In WebSphere MQ Resource Adapter") or
         (desc == "WAS Built In WebSphere MQ Resource Adapter"):
         print "Updating archivePath and classpath of " + ra
         AdminConfig.modify(ra, [['archivePath', "${WAS_INSTALL_ROOT}/installedConnectors/wmq.jmsra.rar"]])
         AdminConfig.unsetAttributes(ra, ['classpath'])
         AdminConfig.modify(ra, [['classpath', "${WAS_INSTALL_ROOT}/installedConnectors/wmq.jmsra.rar"]])
         AdminConfig.save()
      #end if
   #end for
   ```

3. Use the wsadmin tool to run the Jython script that you have just created.

Open a command prompt and navigate to the *profile_root*/bin directory, then enter the following command:

```
wsadmin -lang jython -f convertWMQRA.py
```

4. Stop all servers in the profile.
5. Run the osgiCfgInit command from the *profile_root*/bin directory.

   Note: The osgiCfgInit command resets the class cache used by the OSGi runtime environment.
6. Restart all servers in the profile.

What to do next

If you continue to experience problems after performing the steps described in this topic, and you have previously used the **Update resource adapter...** button on the JMS Provider Settings panel in the administrative console to update the WebSphere MQ resource adapter on any nodes in your environment, it is possible that you are experiencing the issue described in APAR PM10308.

Installing a specific maintenance level of the WebSphere MQ resource adapter

If you require a specific version of the WebSphere MQ resource adapter to be installed and the version you require is unavailable in a WebSphere Application Server fix pack, or an interim fix, you can install the resource adapter using the administrative console.

Before you begin

The minimum version of the WebSphere MQ resource adapter required for this version of WebSphere Application Server is WebSphere MQ Version 7.0.

About this task

Use this task only if you have a requirement to use a specific maintenance level of the WebSphere MQ resource adapter that is unavailable in a WebSphere Application Server fix pack, or an interim fix.

You do not need to use this task if you want to restore the configuration of all servers in your cell to use the latest version of the WebSphere MQ resource adapter contained in the WebSphere Application Server installation. In this case, see “Ensuring that servers use the latest available WebSphere MQ resource adapter maintenance level” on page 725 for further information.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Resources** > **JMS->JMS providers** The JMS providers collection panel opens.
3. Change the scope to the node that you want to update.
4. Select the WebSphere MQ messaging provider entry in the list. The settings panel for this provider opens.
5. Click **Update resource adapter...**.

   Note: **Update resource adapter...** is only available for the WebSphere MQ messaging provider at node scope.
6. Specify the installation path for the resource adapter archive (RAR) file, then click **Next**:
   - If your RAR file is located on the same workstation as your browser, select **Local file system**, and browse to find the file.
   - If your RAR file is located on the server workstation where the application server is installed, select **Remote file system**, and specify the fully qualified path to the file.

7. Review the configuration information that is provided for the RAR file then, when you have finished your review, click **Next**. The following information is displayed for the RAR file:
   - Name
   - Current RAR version
   - New RAR version
   - Scope
   - Any existing copies of the resource adapter. The resource adapters shown with an asterisk (*) are copies of the resource adapter and must also be updated at the same time.
8. Optional: Edit any properties that were added by the new version of the resource adapter. You can also edit these properties after you have completed the update.
   a. From the displayed list, select the resource for which you want to edit the new properties. Only resources with new properties are included in the list.
   b. Edit the resource properties. Use the table that is provided to set the values for new properties of the selected resource.
      - Select **Set for all** to apply the property value to all resources of the same type.
      - Click **Reset to Default** to reset all the properties to the default values that are defined in the RAR file. This resetting of property values only affects the selected resource.
   c. Click **Next**.
9. Review the summary panel and then, when you are satisfied with the configuration settings, click **Finish**. When you click **Finish**, all the configuration changes are saved automatically. To revert to an older version of the resource adapter you must perform the update process again, and specify the older version of the RAR file that you want to revert to.
10. Restart the servers that contain the updated RAR file.

Listing JMS resources for the WebSphere MQ messaging provider

Use the WebSphere Application Server administrative console to list JMS resources for the WebSphere MQ provider, for administrative purposes.

About this task

You use the WebSphere Application Server administrative console to list JMS resources, if you want to view, modify or delete any of the following resources:

- Activation specifications
- Unified connection factories
- Queue connection factories
- Topic connection factories
- Queues
- Topics

When you use the administrative console to locate these resources, two different navigation pathways are available:

- **Provider-centric navigation** lets you view all providers, or just those for a specified scope, then navigate to a specific resource for a specific provider. This is the traditional way of navigating to a resource when you know which provider supports it. Any navigation that starts with **Resources** > **JMS->JMS providers** is provider-centric.
- **Resource-centric navigation** lets you view all resources of a specified type, then navigate to a resource. This is useful if you want to find a resource, but you do not know which provider supports it (you can list all resources of a given type across all scopes, for all providers, in a single panel). Any navigation that follows the pattern **Resources** > **JMS** > ***resource_type*** is resource-centric, where *resource_type* is one of the resource types previously listed.

You can use either of these navigation pathways to locate JMS resources of any type.

Procedure

- Use provider-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console.
  2. In the navigation pane, click **Resources** > **JMS->JMS providers**.

     The JMS providers collection panel is displayed. This lists all currently configured messaging providers across all scopes (you can modify the scope if required).
  3. Select the required JMS provider.

     The settings panel for this provider is displayed. The configuration tab contains a set of links to all the JMS resources owned by this provider.
  4. Click the link for a JMS resource type. For example, click **Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories for this provider.
  5. Select the required queue connection factory.
- Use resource-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console..
  2. In the navigation pane, click **Resources** > **JMS->Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories across all messaging providers.
  3. Select the required queue connection factory.

Results

You can now view and work with the resource properties.

JMS providers collection

Use this panel to list JMS providers, or to select a JMS provider to view or change its configuration properties.

To view this administrative console page, click **Resources** > **JMS->JMS providers**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS providers that are available to WebSphere applications. For each JMS provider in the list, the entry indicates the *scope* level at which JMS resource definitions are visible to applications. You can create the same type of JMS provider at different Scope settings, to offer JMS resources at different levels of visibility to applications.

If you want to manage existing JMS resource definitions, or create a new JMS resource definition, you can select the name of one of the JMS providers in the list.

If you want to define a new third-party JMS provider (that is, a provider other than the default messaging provider or the WebSphere MQ messaging provider), select the Scope setting at which JMS resource definitions are to be visible for that provider, then click **New**.

Name The name by which this JMS provider is known for administrative purposes.

Description
A description of this JMS provider for administrative purposes.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Activation specification collection

A JMS activation specification is associated with one or more message-driven beans and provides the configuration necessary for them to receive messages. The default messaging provider and the WebSphere MQ messaging provider both support use of activation specifications.

In the administrative console, to view this page click **Resources** > **JMS->Activation specifications**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

This page lists the JMS activation specifications that are available to WebSphere applications at the scope indicated by the **Scope** field.

Name The display name of each activation specification instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each activation specification instance.

Provider
The messaging provider that supports each activation specification instance. This is either the default messaging provider (service integration) or the WebSphere MQ messaging provider.

Description
An optional description of each activation specification instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.

Delete	Delete the selected items.

Connection factory collection

A JMS connection factory is used to create connections to the associated messaging provider of JMS destinations, for both point-to-point and publish/subscribe messaging. Use connection factory administrative objects to manage JMS connection factories for the default messaging provider, the WebSphere MQ messaging provider or a third-party messaging provider.

In the administrative console, to view this page click **Resources** > **JMS->Connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS connection factory is used to create connections to JMS destinations. When an application needs a JMS connection, an instance can be created by the factory of the messaging provider named in the **Provider** column of the list.

This type of connection factory is for applications that use the JMS 1.1 domain-independent interfaces (referred to as the “common interfaces” in the JMS specification).

This type of JMS connection factory can also be used by the domain-specific (queue and topic) interfaces, as used in JMS 1.0.2, so applications can still use those interfaces without the need for you to create a domain-specific connection factory, such as a queue connection factory.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The display name of each connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each connection factory instance.

Provider
The messaging provider that supports each connection factory instance. This is the default messaging provider (service integration), the WebSphere MQ messaging provider or a third-party messaging provider.

Description
An optional description of each connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Queue connection factory collection

A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for point-to-point messaging.

In the administrative console, to view this page click **Resources** > **JMS->Queue connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS queue connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS queue connection factory is used to create connections to JMS destinations. When an application needs a JMS queue connection, an instance can be created by the factory for the JMS provider that is named in the **Provider** column of the list.

This type of connection factory is for applications that use the JMS 1.0.2 queue-specific interfaces.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The display name of each queue connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each queue connection factory instance.

Provider
The messaging provider that supports each queue connection factory instance.

Description
An optional description of each queue connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Topic connection factory collection

A JMS topic connection factory is used to create connections to the associated messaging provider of JMS topic destinations, for publish and subscribe messaging.

In the administrative console, to view this page click **Resources** > **JMS->Topic connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS topic connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS connection factory is used to create connections to JMS destinations. When an application needs a JMS connection, an instance can be created by the factory for the JMS provider that is named in the Provider column of the list.

This type of connection factory is for applications that use the JMS 1.0.2 topic-specific interfaces.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each topic connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each topic connection factory instance.

Provider
The messaging provider that supports each topic connection factory instance.

Description
An optional description of each topic connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Queue collection

A JMS queue destination is used for point-to-point messaging. Use this panel to create or delete queue destinations, or to select a queue destination to view or change its configuration properties.

In the administrative console, to view this page click **Resources** > **JMS->Queues**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS queue destinations that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

Use a queue destination to manage JMS queues for the JMS provider that is named in the **Provider** column of the list. Connections to the queue are created by a unified connection factory or queue connection factory for that JMS provider.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each queue destination instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each queue destination instance.

Provider
The messaging provider that supports each queue destination instance.

Description
An optional description of each queue destination instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Topic collection

A JMS topic destination is used for publish and subscribe messaging. Use this panel to create or delete topic destinations, or to select a topic destination to view or change its configuration properties.

In the administrative console, to view this page click **Resources** > **JMS->Topics**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS topic destinations that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

Use a topic destination to manage JMS topics for the JMS provider that is named in the **Provider** column of the list. Connections to the topic are created by a unified connection factory or topic connection factory for that JMS provider.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see "Disabling WebSphere MQ functionality in WebSphere Application Server" on page 868.

General properties

Name The display name of each topic destination instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each topic destination instance.

Provider
The messaging provider that supports each topic destination instance.

Description
An optional description of each topic destination instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Configuring JMS resources for the WebSphere MQ messaging provider

Use the WebSphere Application Server administrative console to configure activation specifications, connection factories and destinations for the WebSphere MQ JMS provider.

Before you begin

This task assumes that you are working in a mixed WebSphere Application Server and WebSphere MQ environment, and that you have decided to use the WebSphere MQ messaging provider to handle JMS messaging between the two systems. If your business uses WebSphere MQ, and you want to integrate WebSphere Application Server messaging applications into a predominately WebSphere MQ network, the WebSphere MQ messaging provider is the natural choice. However, there can be benefits in using another provider. If you are not sure which provider combination is best suited to your needs, see Choosing messaging providers for a mixed environment.

You can configure JMS resources for the WebSphere MQ messaging provider through the administrative console as described in this task, or you can configure JMS resources for the WebSphere MQ messaging provider through the WebSphere MQ administrative commands.

About this task

Using the administrative console, you can set the scope of the WebSphere MQ messaging provider to restrict the range of resources that are displayed:

- If you set the scope to contain only WebSphere Application Server Version 6 or Version 7.0 or later nodes, you can configure JMS 1.1 resources and properties. This includes unified JMS connection factories for use by both point-to-point and publish/subscribe JMS 1.1 applications. With JMS 1.1, this approach is preferred to the domain-specific queue connection factory and topic connection factory.
- If you set the scope to contain only WebSphere Application Server Version 7.0 or later nodes, you can also configure JMS activation specifications.
- If you set the scope to a WebSphere Application Server Version 5 node, you can only configure domain-specific JMS resources, and the subset of properties that apply to WebSphere Application Server Version 5.

Note:

There are two ways of specifying the information needed by WebSphere MQ messaging provider messaging resources so that they can connect to a WebSphere MQ queue manager. It can either be specified manually, or by providing the WebSphere MQ messaging provider resource with a uniform resource locator (URL) that points to a client channel definition table (CCDT).

A CCDT is a binary file that contains information about how to create a client connection channel to one or more queue managers. The file contains information such as the hostname, port, and name of the target queue manager, as well as more advanced configuration information like the SSL attributes that should be used.

Creating WebSphere MQ messaging provider resources using CCDTs provides the following benefits:

- Flexibility, because client connection channel information is contained in a single place. If any of the information changes, such as the host name of the machine on which the WebSphere MQ queue manager resides, only the CCDT needs to be updated. When it is updated, all WebSphere MQ messaging provider resources that make use of the CCDT pick up the change.
- Reliability, because less information is needed for a CCDT there is a reduced chance of configuration errors. When using a CCDT to enter connection information, all that is required are the CCDT URL and an optional queue manager name. If you configure a WebSphere MQ messaging provider resource manually, much more information is required -- especially if you are configuring SSL.

For further information about generating a CCDT, see the WebSphere MQ information center.

Maintenance note: The WebSphere MQ messaging provider uses code provided by the WebSphere MQ resource adapter, which is automatically installed as part of the product.

Procedure

- “Creating an activation specification for the WebSphere MQ messaging provider” on page 736
- “Configuring an activation specification for the WebSphere MQ messaging provider” on page 738
- “Migrating a listener port to an activation specification for use with the WebSphere MQ messaging provider” on page 758
- “Creating a connection factory for the WebSphere MQ messaging provider” on page 759

- “Configuring a unified connection factory for the WebSphere MQ messaging provider” on page 761
- “Configuring a queue connection factory for the WebSphere MQ messaging provider” on page 789
- “Configuring a topic connection factory for the WebSphere MQ messaging provider” on page 813
- “Configuring a queue for the WebSphere MQ messaging provider” on page 840
- “Configuring a topic for the WebSphere MQ messaging provider” on page 852
- “Configuring custom properties for WebSphere MQ messaging provider JMS resources” on page 862

Creating an activation specification for the WebSphere MQ messaging provider

Use this task to create an activation specification for use with the WebSphere MQ messaging provider.

About this task

To create an activation specification for use with the WebSphere MQ messaging provider, use the administrative console to complete the following steps. You can choose either to enter all the required connection information using the Create WebSphere MQ JMS resource wizard or to use a client channel definition table (CCDT) to establish a connection to the WebSphere MQ messaging provider.

For information about modifying an existing activation specification, see the related tasks.

Procedure

1. In the navigation pane, click **Resources** > **JMS** > **Activation specifications**.
2. Select the **Scope** setting corresponding to the scope of the activation specification that you want to create.
3. Click **New** in the content pane.
4. Select **WebSphere MQ messaging provider**, then click **OK** to start the Create WebSphere MQ JMS resource wizard.
5. On the “Configure basic attributes” page, enter the following information, then click **Next**.

   Name The name by which this activation specification is known for administrative purposes within WebSphere Application Server.

   JNDI name
   The name that is used to bind this activation specification into the JNDI namespace.

   Description
   Optional. A description of this activation specification for administrative purposes within WebSphere Application Server.
6. On the “Specify MDB destination data” page, enter the following information, then click **Next**.

   Destination JNDI name
   The JNDI name for the JMS destination from which messages are delivered to a message-driven bean (MDB) that is configured to use this activation specification.

   Message selector
   Optional. A message selector expression specifying which messages are to be delivered.

   Destination type
   The type of destination (queue or topic) from which to consume messages.
7. On the “Select connection method” page, choose how you want to specify the connection details for to the WebSphere MQ messaging provider by selecting one of the following options, then click **Next**.

   Enter all the required information into this wizard
   Choosing this option takes you to the “Supply queue manager details” page (see step 8), where you can start entering the connection details using the Create WebSphere MQ JMS resource wizard.

Use a client channel definition table

Choosing this option takes you to the "Specify client channel definition table" page (see step 9), where you can enter details of the client channel definition table that you want to use.

8. If you have chosen to enter all the required information using the wizard, complete the following steps:
   a. On the "Supply queue manager details" page, enter the name of the queue manager or queue-sharing group that you want to connect to then click **Next**.
   b. On the "Enter connection details" page, specify the details for the connection, then click **Next** to continue to the "Test connection" page. You can choose either to enter host and port information separately, or enter host and port information in the form of a connection name list. You must only choose the option to use a connection name list if you are creating a connection to a multi-instance queue manager. You must not use this option for connections to non-multi-instance queue managers as that can result in transaction integrity issues.

      Transport

      Optional. The WebSphere MQ transport type for the connection. The option that you select is used to determine the exact mechanisms used to connect to WebSphere MQ. If you are configuring a transport type of *bindings, then client* or *bindings*, see "Configuring the WebSphere MQ messaging provider with native libraries information" on page 723 for more information.

      Enter host and port information in the form of separate hostname and port values

      This radio button is selected by default and enables the **Hostname** and **Port** fields and disables the **Connection name list** field. Leave this radio button selected if you want to enter host and port information in the form of separate hostname and port values.

      Hostname

      The host name, IPv4, or IPv6 address of the WebSphere MQ queue manager to connect to. Complete this field if you have selected **Enter host and port information in the form of separate hostname and port values**.

      Port Optional. The port number on which WebSphere MQ is listening. If you do not specify a value for the port number, the default value of 1414 is used.Complete this field if you have selected **Enter host and port information in the form of separate hostname and port values**.

      Enter host and port information in the form of a connection name list

      This radio button is cleared by default and, if selected, disables the **Hostname** and **Port** fields and enables the **Connection name list** field.
      - Click this radio button to select it if you want to enter host and port information in the form of a connection name list.
      - Leave this radio button cleared if you want to enter host and port information in the form of separate host name and port values.

      Connection name list

      The connection name list specifying the host name and port details to use when you want the connection factory to connect to a multi-instance queue manager. Complete this field if you have selected **Enter host and port information in the form of a connection name list**. Enter the host name and port details in the following form:

      `host[(port)][,host(port)]`

      host must be a valid TCP/IP host name or IPv4 or IPv6 address.

      port must be an integer value in the range 1 - 65536 (inclusive). The port information is optional, and if not specified, defaults to 1414.

      For example: `localhost(1234),remotehost1(1234),remotehost2`

Server connection channel
Optional. The WebSphere MQ server connection channel name used when connecting to WebSphere MQ queue manager or queue-sharing group.

9. If you are using a channel client definition table to establish a connection to the WebSphere MQ messaging provider, complete the following fields on the “Specify client channel definition table” page, then click **Next** to continue to the “Test connection” page.

   Client channel definition table URL
   The URL to the client channel definition table that you want to use when connecting to WebSphere MQ.

   Queue manager
   Optional. The name of the queue manager to be used to select one or more entries from the CCDT.

10. Optional: On the “Test connection” page, if you want to test establishing the connection, click **Test connection**. This test can take several seconds to perform.
11. On the “Summary” page, complete the creation of the new activation specification by clicking **Finish**.
12. Stop then restart the application server.

JMS resource provider selection panel

Select the messaging provider with which to create this JMS activation specification, connection factory or destination.

You select the scope setting on an earlier page. The choice of JMS provider depends on the scope that you selected. You might see a choice such as the following list:

- Default messaging provider.

  Select this option if you want the JMS resource to be provided by the service integration bus, as part of WebSphere Application Server.
- WebSphere MQ messaging provider

  Select this option if you want the JMS resource to be provided by WebSphere MQ. You must have installed and configured a WebSphere MQ network in order to use this provider.
- “My JMS provider”

  Select this option if you want the JMS resource to be provided by a third-party JMS provider. This option is only available if you have installed and configured a third-party provider. The name that is displayed (for example“My JMS provider”) is the name you gave to the provider when you installed and configured it.
- V5 default messaging provider

  Select this option if you want to manage the JMS connection factories and destinations for JMS applications that were designed to work with WebSphere Application Server Version 5.1, and are still running on a Version 5.1 or Version 6 node that is part of a Version 7.0 or later mixed cell.

Configuring an activation specification for the WebSphere MQ messaging provider

Configure a JMS activation specification to enable a message-driven bean (MDB) to communicate with the WebSphere MQ messaging provider.

About this task

To view or change the configuration of an activation specification for use with WebSphere MQ, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Resources** > **JMS->Activation specifications**. A list of existing activation specifications, with a summary of their properties, is displayed in an "Activation specification collection" on page 729 form.
2. Select the **Scope** setting corresponding to the scope of the activation specifications that you want to view or change.
3. Select the name of the activation specification that you want to view or change. Configuration details for the activation specification are displayed in an "WebSphere MQ messaging provider activation specification settings" form.
4. Under **General Properties** make modifications as necessary.

   For information about each of the available fields, see "WebSphere MQ messaging provider activation specification settings."

   Note: There are four WebSphere MQ connection properties that are used to configure the WebSphere MQ resource adapter used by the WebSphere MQ messaging provider. These properties affect the connection pool that is used by activation specifications:
   - maxConnections
   - connectionConcurrency
   - reconnectionRetryCount
   - reconnectionRetryInterval

   For more information about these four properties, and how to configure them, see "Resource adapter properties" on page 721.
5. Click **Apply** to save the configuration.
6. Optional: Click **Advanced properties** to display or change the list of advanced properties of your activation specification.
7. Optional: Click **Broker properties** to display or change the list of broker properties of your activation specification.
8. Optional: Click **Custom properties** to display or change the list of custom properties of your activation specification. For example, you would use this option to set the custom property WAS_EndpointInitialState for an activation specification. WAS_EndpointInitialState determines whether or not message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.
9. Optional: Click **Client transport properties** to display or change the list of client transport properties of your activation specification. This link is only present on the explicitly-defined variation of this panel, where you enter all information required to connect to WebSphere MQ. This link does not appear for the CCDT-based variation.
10. Click **OK**.
11. Save your changes to the master configuration.
12. To have the changed configuration take effect, stop then restart the application server.

WebSphere MQ messaging provider activation specification settings

Use this panel to view or change the configuration properties of the selected activation specification for use with the WebSphere MQ messaging provider. These configuration properties control how connections are created to associated queues and topics.

To view WebSphere MQ activation specification settings, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->Activation specifications** to display existing activation specifications.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the activation specifications are defined. This restricts the set of activation specifications displayed.

3. Select the name of the activation specification that you want to work with.

Under General Properties there are five groups of properties:
- "Administration"
- "Connection" on page 741
- "Destination" on page 747
- "Advanced" on page 748
- "Security settings" on page 748

Make any required changes to the Administration, Connection, Destination, Advanced, and Security settings groups of properties, and then click **Apply** to save the configuration before, in the content pane under Additional Properties, you click any of the following links:
- **Advanced properties** to display or change the advanced properties of your WebSphere MQ activation specification.
- **Broker properties** to display or change the broker properties of your WebSphere MQ activation specification.
- **Custom properties** to display or change the custom properties of your WebSphere MQ activation specification. For example, you would use this option to set the custom property WAS_EndpointInitialState for an activation specification. WAS_EndpointInitialState determines whether or not message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.
- **Client transport properties** to display or change the client transport properties of your WebSphere MQ activation specification. If the selected activation specification was not created using a Client Channel Definition Table (CCDT), follow this link to enter all the information required to connect to WebSphere MQ. If the selected activation specification was created using a CCDT, you do not need to supply the client transport properties, and so the link is absent.

Under Related Items, you can click **JAAS - J2C authentication data** to configure authentication information for use with the activation specification.

You can also specify the **-localAddress** property by using the createWMQActivationSpec WebSphere MQ administrative command.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information, see the *Using Java* and *System Administration* sections of the WebSphere MQ information center.

If WebSphere MQ functionality has been disabled at a scope that affects this WebSphere MQ messaging provider resource, then an informational message indicating that WebSphere MQ has been disabled appears. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see "Disabling WebSphere MQ functionality in WebSphere Application Server" on page 868.

A WebSphere MQ activation specification has the following properties.

Administration:

Scope:

The level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the WebSphere MQ activation specification collection panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Provider:

The JMS provider assigned when the activation specification is created.

For all activation specifications created using this panel, the provider is the WebSphere MQ messaging provider.

The provider is displayed for information only.

Data type String

Name:

The name by which this activation specification is known for administrative purposes within WebSphere Application Server.

Data type String
Range The name must be unique within the set of activation specifications defined to the cell.

JNDI name:

The JNDI name that is used to bind the activation specification into the JNDI namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

Data type String

Description:

A description of this activation specification for administrative purposes within WebSphere Application Server.

Data type String
Default Null

Connection:

The information required to configure a connection depends on whether the selected activation specification was created using a Client Channel Definition Table (CCDT).

If the selected activation specification was created using a CCDT, only the following properties are displayed:

- Client channel definition table URL
- Queue manager
- SSL configuration

If the selected activation specification was not created using a CCDT, the following properties are displayed:

- Queue manager
- Transport
- If **Enter host and port information in the form of separate host and port values** is selected, the connection name list property cannot be used and the following properties can be used:
  - Host name
  - Port
- If **Enter host and port information in the form of a connection name list** is selected, the connection name list property can be used and the following properties cannot be used:
  - Host name
  - Port
- Server connection channel
- If you clear the check box for the **Use SSL to secure communication with Websphere MQ** property, the following properties cannot be used:
  - Centrally managed
  - Specific configuration
  - SSL configuration

For more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *WebSphere MQ Using Java* section of the WebSphere MQ information center.

Note: There are four WebSphere MQ connection properties that are used to configure the WebSphere MQ resource adapter used by the WebSphere MQ messaging provider. These properties affect the connection pool that is used by activation specifications:

- maxConnections
- connectionConcurrency
- reconnectionRetryCount
- reconnectionRetryInterval

For more information about these four properties, and how to configure them, see “Resource adapter properties” on page 721.

Client channel definition table URL:

A URL that specifies the location of a WebSphere MQ CCDT.

Data type String

Queue manager:

If the specified activation specification is based on a CCDT, this property is used to select an entry in the CCDT. Otherwise, this property specifies the name of the queue manager or queue-sharing group to connect to. A connection is established to the specified WebSphere MQ resource to receive messages.

Data type String

Range
If this activation specification is not based on a CCDT, the value must be a valid queue manager name.

If this activation specification is based on a CCDT, the value must be one of the following:

- A valid queue manager name
- An asterisk (*) followed by the name of a queue manager group[1]
- An asterisk (*)
- Blank[1]

[1]When you specify the value of the Queue manager property in this form in combination with a CCDT, individual connections established by using the activation specification might connect to different queue managers. Selection from multiple queue managers occurs when the CCDT contains multiple client connection channel definitions with a matching queue manager name (QMNAME) parameter, and these connection channel definitions define the network connection details of different queue managers.

If the specified connection factory is based on a CCDT, and the CCDT can select from more than one queue manager, you might not be able to recover global transactions. Therefore, for connection factories that specify a CCDT, you have two alternatives:

- Avoid any ambiguity about the target queue manager when specifying the Queue manager property, which means that the specified value of this property must not include an asterisk (*).
- Avoid using the resources with applications that enlist in global transactions.

Transport:

The WebSphere MQ transport type for the connection. The transport type is used to determine the exact mechanisms used to connect to WebSphere MQ.

Data type Drop-down list

Default **Bindings, then client**

Range

Client Use a TCP/IP-based network connection to communicate with the WebSphere MQ queue manager.

Bindings, then client
Attempt a bindings mode connection to the queue manager. If this is not possible, revert to the client transport.

Bindings
Establish a cross-memory connection to a queue manager running on the same node. The following Client Transport Mode properties are disabled:
- Host name
- Port
- Connection name list
- Server connection channel

For more information about configuring a transport type of *Bindings, then client* or *Bindings*, refer to "Configuring the WebSphere MQ messaging provider with native libraries information" on page 723.

Enter host and port information in the form of separate host and port values:

If this radio button is selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the host name and port properties.

Selecting this option enables the host name and port properties, and disables the connection name list property. To enter connection name list information, click **Enter host and port information in the form of a connection name list**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type Radio button

Hostname:

The host name, IPv4, or IPv6 address of the WebSphere MQ queue manager to connect to.

Data type String

Port:

The port number on which WebSphere MQ is listening.

Data type Integer
Default 1414
Range The value must be in the range 1 to 65536 (inclusive).

Enter host and port information in the form of a connection name list:

If this radio button is selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the connection name list property.

Connection name lists can be used to connect to a single queue manager or to a multi-instance queue manager. For more information on using a multi-instance queue manager, see the WebSphere MQ information centre. Selecting this option enables the connection name list property and disables the host name and port properties. To enter separate host and port information, click **Enter host and port information in the form of separate host and port values**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type	Radio button

Connection name list:

The connection name list specifying the host name and port details to use when you want the activation specification to connect to a multi-instance queue manager.

This property must only be used to allow connection to a multi-instance queue manager. It must not be used to allow connections to non-multi-instance queue managers as that can result in transaction integrity issues.

Data type	String
Default	Unset
Range	This field must be set to a string in the following form: *host*[(*port*)][,*host*(*port*)] The port information is optional, and if not specified, defaults to 1414. *host* must be a valid TCP/IP host name or IPv4 or IPv6 address. *port* must be an integer value in the range 1 to 65536 (inclusive). For example: `localhost(1234),remotehost1(1234),remotehost2`

When the connection name list property is specified, the host name or port properties are automatically set to the host name and port number of the first entry in the connection name list. So if you specified `localhost(1234),remotehost1(1234),remotehost2`, the host name would be set to localhost and port would be set to 1234.

This property is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

When used in a mixed cell environment, the information in the connection name list property, for cell scope activation specifications, is available to WebSphere Application Server Version 7.0 nodes. The exact behaviour depends on the fix pack level of the node:

- For nodes running at a fix pack level of WebSphere Application Server Version 7.0 Fix Pack 7 or later, the connection name list property can be used to connect to multi-instance queue managers.
- For nodes running at a fix pack level earlier than Version 7.0, the connection name list property is not recognized, and a warning message similar to the following example is output:

```
[29/09/10 12:15:27:468 BST] 00000018 J2CUtilityCla W
J2CA0008W: Class com.ibm.mq.connector.inbound.ActivationSpecImpl used by resource
cells/L3A3316Node01Cell/resources.xml#J2CResourceAdapter_1284547647859 did not contain
method setConnectionNameList. Processing continued.
```

In this case the information in the host name and port properties are used to connect to a queue manager.

Server connection channel:

The WebSphere MQ server connection channel name used when connecting to WebSphere MQ.

Data type	String
Default	SYSTEM.DEF.SVRCONN
Range	The value must be a server connection channel defined to the WebSphere MQ queue manager that is being connected to.

Use SSL to secure communications with WebSphere MQ:

This option determines whether the SSL (Secure Sockets Layer) protocol is used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

When using a WebSphere MQ messaging provider activation specification in the application server environment, the application server manages SSL configuration. To change SSL configuration parameters, use the administrative console to navigate to the **Security** > **SSL certificate and key management** panel.

You can only use one cipher suite in the SSL configuration for a WebSphere MQ messaging provider activation specification . If you specify more than one cipher suite, only the first one is used.

Data type	Check box. If this check box is cleared, the following SSL properties are disabled: • Centrally managed • Specific configuration • SSL configuration

Centrally managed:

When the SSL protocol is used to communicate with WebSphere MQ, select this radio button to specify that the SSL configuration is taken from the centrally managed WebSphere Application Server SSL configuration.

When you select this radio button, the host name and port attributes from the WebSphere MQ messaging provider activation specification are used to select an appropriate SSL configuration. If host and port information has been supplied to the activation specification by a connection name list this means that the host name and port information of the first element in the list are used to select an appropriate SSL configuration. To provide the SSL configuration which will be matched to the activation specification, see the Dynamic outbound endpoint SSL configuration settings topic listed under related reference.

Data type	Radio button

Specific configuration:

Select this radio button when you want to specify a particular SSL configuration for use when SSL is to be used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

Data type	Radio button

SSL configuration:

The SSL configuration to use when SSL is to be used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

This property is disabled if the **Centrally managed** radio button is selected and the WebSphere MQ messaging provider resource has been explicitly defined.

This property is always enabled if the WebSphere MQ messaging provider resource is based on a CCDT.

If this WebSphere MQ messaging provider resource is based on a CCDT, this parameter is only used if the relevant entries in the CCDT have been configured to use SSL.

Additionally, if an SSL configuration of `none` is selected, the default centrally managed WebSphere Application Server SSL configuration for the WebSphere MQ messaging provider is used.

Data type Drop-down list

Destination:

Note: The property WAS_EndpointInitialState is a custom property. This property determines whether or not message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination. To set custom properties, when you have completed your changes in this content pane and clicked **Apply** to save the configuration, then click the link **Custom properties** in the content pane under Additional Properties.

Destination JNDI name:

The JNDI name for the JMS destination from which messages are consumed for delivery to a message-driven bean (MDB) that is configured to use this activation specification.

Data type String

Message selector:

A message selector expression specifying which messages are to be delivered.

Data type String

Destination type:

The type of destination (queue or topic) from which to consume messages.

Data type Drop-down list

Range

Queue The Destination JNDI name refers to a JMS destination that is a queue.

Topic The Destination JNDI name refers to a JMS destination that is a topic.

Durable subscription:

An option to specify whether a durable or nondurable subscription is used to deliver messages to an MDB subscribing to the topic.

Data type Check box

Default Cleared (nondurable)

Range

Cleared
Nondurable.
Selected
Durable.

Subscription name:

The name of a durable subscription. This is available only when the **Durable subscription** check box is selected.

Data type String

Advanced:

Client ID:

The client identifier to specify when connecting to the WebSphere MQ messaging provider.

Data type String

Allow cloned durable subscriptions:

An option that determines whether multiple instances of a durable subscription can be accessed concurrently by different servers.

Data type Check box
Default Cleared
Range

Selected
Multiple instances of a durable subscription can be accessed concurrently by different servers.
Cleared
Multiple instances of a durable subscription cannot be accessed concurrently by different servers.

Provider version:

The WebSphere MQ messaging provider version. This is used to determine whether to connect to a particular version of a queue manager. It is also used to determine the type of functions required by the client.

Data type String
Range
The value entered must either be the empty string or be in one of the following formats:

n.n.n.n

n.n.n

n.n

n

where n is a numeric value greater than or equal to zero.

For example 6.0.0.0.

Security settings:

Authentication alias:

The user name and password to use when connecting to WebSphere MQ.

Data type	Drop-down list
Default	(none)
Range	All authentication aliases defined to the cell and the value "(none)", which specifies that no credentials are passed to WebSphere MQ.

WebSphere MQ messaging provider activation specification advanced properties:

Use this panel to view or change the advanced properties of the selected activation specification for use with the WebSphere MQ messaging provider. These advanced properties control the behavior of connections made to WebSphere MQ messaging provider destinations.

To view WebSphere MQ activation specification advanced properties, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->Activation specifications** to display existing activation specifications.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the activation specifications are defined. This restricts the set of activation specifications displayed.
3. Select the name of the activation specification that you want to work with.
4. In the content pane, under Additional properties, click **Advanced properties** to view a list of the advanced properties of the WebSphere MQ activation specification.

Under General Properties there are four groups of properties:

- Message compression
- Connection consumer
- Message format
- Additional

Make any required changes to these groups and then click **Apply** to return to the activation specification.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the WebSphere MQ *Using Java* section of the WebSphere MQ library.

A WebSphere MQ activation specification has the following advanced properties.

Compress message headers:

An option that enables the compression of message headers.

Data type	Check box
Default	Cleared
Range	**Cleared** Do not compress message headers. **Selected** Compress message headers.

Compression algorithm for message payloads:

The compression algorithm that is used to compress message payloads.

Data type Drop-down list
Default NONE
Range
RLE Message data compression is performed using run-length encoding.
ZLIBFAST Message data compression is performed using ZLIB encoding with speed prioritized.
ZLIBHIGH Message data compression is performed using ZLIB encoding with compression prioritized.
NONE No message data compression is performed.

Retain messages, even if no matching consumer is available:

An option that determines whether messages for which there is no matching consumer are retained on the input queue or dealt with according with their disposition options.

Data type Check box
Default Selected
Range
Cleared Do not retain messages.
Selected Retain messages.

Rescan interval:

When using a WebSphere MQ version 6 queue manager (or WebSphere MQ version 5.3 for z/OS), this setting configures the mechanism that is used to dispatch messages to JMS asynchronous consumers.

This setting is used when the set of WebSphere MQ queues that is being asynchronously consumed from exceeds the number of threads that are available internally to synchronously get messages from the WebSphere MQ queue. The setting determines how long a thread retrieves messages from a WebSphere MQ queue before switching to consume messages from another WebSphere MQ queue in the set.

Data type Integer
Units Milliseconds
Default 5000
Range A value greater than zero.

Maximum server sessions:

The maximum number of server sessions in the server session pool that is used by the connection consumer.

Data type Integer
Default 10
Range A value greater than zero.

Server session pool timeout:

The period of time, in milliseconds, that an unused server session is held open in the server session pool before being closed due to inactivity.

Data type	Integer
Units	Milliseconds
Default	300,000
Range	A value greater than zero.

Start timeout:

The period of time, in milliseconds, within which delivery of a message to a message-driven bean (MDB) must start after the work to deliver the message has been scheduled. If this period of time elapses, the message is rolled back onto the queue.

Data type	Integer
Units	Milliseconds
Default	10,000
Range	A value greater than zero.

Coded character set identifier:

The character set to use when you are encoding strings in the message.

Data type	Integer
Default	819
Range	A value greater than zero. The value must be one of the CCSIDs supported by WebSphere MQ.

For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *WebSphere MQ System Administration* and the *WebSphere MQ Application Programming Reference* sections of the WebSphere MQ library.

Fail JMS method calls if the WebSphere MQ queue manager is quiescing:

An option that enables selected JMS operations to fail when the queue manager is put into a quiescing state. Selecting this option enables the queue manager to quiesce successfully and shut down.

Data type	Check box
Default	Selected
Range	**Cleared** Do not fail JMS operations if the queue manager is quiescing. **Selected** Fail JMS operations if the queue manager is quiescing.

Stop endpoint if message delivery fails:

An option that determines whether message delivery is suspended to a failing endpoint.

Data type	Check box
Default	Selected

Range

Cleared
Message delivery is not suspended to a failing endpoint.

Selected
Message delivery is suspended to a failing endpoint when the value for the **Number of sequential delivery failures before suspending endpoint** is exceeded.

Number of sequential delivery failures before suspending endpoint:

The number of sequential message delivery failures to an endpoint that are allowed before message delivery to that endpoint is suspended. This property is enabled only when **Suspend message delivery to failing endpoints** is selected.

Data type	Integer
Default	0
Range	A value greater than or equal to zero.

WebSphere MQ messaging provider activation specification broker properties:

Use this panel to view or change the broker settings of the selected activation specification for use with the WebSphere MQ messaging provider. These broker settings determine how the WebSphere MQ messaging provider interacts with a broker for the purposes of publishing messages and subscribing to topics. Updates to the settings take effect when the server is restarted.

To view WebSphere MQ activation specification broker properties, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Activation specifications** to display existing activation specifications.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the activation specifications are defined. This restricts the set of activation specifications displayed.
3. Select the name of the activation specification that you want to work with.
4. In the content pane under Additional properties, click **Broker properties** to view a list of the broker properties of the WebSphere MQ activation specification.

Under General Properties there are four groups of properties:
- Queues
- Capabilities
- Tuning
- Additional

Make any required changes to these groups and then click **Apply** to return to the activation specification.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the WebSphere MQ *Using Java* section of the WebSphere MQ library.

A WebSphere MQ activation specification has the following broker properties.

Broker control queue:

The queue to which broker control messages are sent.

Data type String
Default SYSTEM.BROKER.CONTROL.QUEUE

Broker subscriber queue:

The queue from which subscription messages are received.

Data type String
Default SYSTEM.JMS.ND.SUBSCRIBER.QUEUE

Broker connection consumer subscription queue:

The queue to receive subscription messages for connection consumers from.

Data type String
Default SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE

Broker connection consumer durable subscription queue:

The queue to receive subscription messages for durable connection consumers from.

Data type String
Default SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE

Version:

The version of the broker that is used. This determines some of the capabilities that the broker is assumed to have. For example, whether to use an RFH version 1 or version 2 header in publications.

Data type Radio button
Default Version 1 broker
Range

Version 1 broker
Message selection cannot be specified.

Version 2 broker
Message selection can be specified. If you select this option, you must also complete **Specify where message selection occurs**.

Specify where message selection occurs:

The process in which message selection is performed. This property is enabled only if **Version 2 broker** was selected.

Data type Drop-down list
Default CLIENT

Range

CLIENT
Message selection is performed in the application server process.

BROKER
Message selection is performed in the broker process.

Subscription store:

The process for tracking subscriptions.

Data type Drop-down list
Default MIGRATE
Range

MIGRATE
Any information that is held on queues is migrated to the broker mechanism for persisting subscription information. If subscription information is already persisted using the broker mechanism then specifying a value of `Migrate` is equivalent to specifying a value of `Broker`.

BROKER
Internal broker mechanisms are used to track subscription information.

QUEUE
A designated WebSphere MQ queue is used to record information about current subscriptions.

Durable subscription state refresh interval:

How often a long running transaction is recreated and used to clean up durable subscriptions, for some versions of the queue manager.

Data type Integer
Default 60000
Range Any positive integer

Subscription cleanup level:

How aggressively messages are cleaned up if the subscriber that is expected to consume the messages terminates unexpectedly.

Data type Drop-down list
Default SAFE

Range

SAFE A conservative algorithm is used to clean up subscriptions.

ASPROP
The cleanup algorithm is determined by a system property.

NONE No cleanup of subscriptions is performed.

STRONG
An aggressive algorithm is used to clean up subscriptions.

Subscription cleanup interval:

How often to check for orphaned subscriptions and clean up messages.

Data type Integer
Default 3600000
Range Any positive integer

Subscription wildcard format:

The wildcard format used for subscribing to more than one topic in a topic hierarchy.

Data type Drop-down list
Default character wildcards
Range

character wildcards
You can use an asterisk (*) or question mark (?) to represent characters, or strings of characters, in a topic name.

* is interpreted as matching many characters.

? is interpreted as matching a single character.

topic level wildcards
You can use a plus sign (+) or number sign (#) to represent topics in a multilevel topic hierarchy.

+ is interpreted as matching a single topic name.

is interpreted as matching many topics in the hierarchy. / is used to delimit topics.

Optimize for sparse subscription patterns:

An option to specify whether this activation specification is anticipated to receive a high proportion of messages that match its selection criteria. This information can be used to optimize message delivery.

Data type Check box
Default Cleared
Range

Cleared
Subscriptions frequently receive matching messages.

Selected
Subscriptions do not frequently receive matching messages.

Broker queue manager:

The name of the queue manager that is running the broker, if it is not the same as the queue manager to which the activation specification connects.

Data type	String
Default	The queue manager name that was specified in the activation specification.

WebSphere MQ messaging provider activation specification client transport properties:

Use this panel to view or change the client transport properties of an activation specification for use with the WebSphere MQ messaging provider. These properties affect how a client connection is established with a WebSphere MQ queue manager or queue-sharing group. Updates to the properties take effect when the server is restarted.

To view WebSphere MQ activation specification client transport properties, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->Activation specifications** to display existing activation specifications.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the activation specifications are defined. This restricts the set of activation specifications displayed.
3. Select the name of the activation specification that you want to work with.
4. In the content pane, under Additional properties, click **Client transport properties** to display a list of the client transport properties of the WebSphere MQ activation specification.

Under General Properties there are two groups of properties:

- Additional SSL settings (for more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *Using Java* section of the WebSphere MQ information center)
- Channel exits

Make any required changes to these groups and then click **Apply** to return to the activation specification.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ activation specification has the following client transport properties:

Certificate revocation list:

A list of LDAP URLs pointing to LDAP repositories of SSL certificates that might have been revoked.

Data type	String
Default	No certificate revocation list
Range	The value must be a space-separated list of LDAP URLs.

Reset count:

The total number of bytes to transfer over an SSL connection before renegotiating the symmetric encryption keys used to secure the connection.

Data type Integer
Default 0 (do not renegotiate)
Range The value must be in the range 0 through 999,999,999 (inclusive).

Peer name:

A name (possibly including wildcards) that must match the distinguished name of the peer SSL certificate for a connection to be established.

Data type String
Default Do not check the distinguished name of the peer certificate.
Range Validated using the rules for a WebSphere MQ SSLPEER channel parameter.

Receive exit or exits:

A comma-separated list of Java class names corresponding to receive exits to be loaded.

Data type String

Receive exit initialization data:

Initialization data to be passed to the receive exit.

Data type String

Send exit or exits:

A comma-separated list of Java class names corresponding to send exits to be loaded.

Data type String

Send exit initialization data:

Initialization data to be passed to the send exit.

Data type String

Security exit:

A Java class name corresponding to the security exit to be loaded.

Data type String

Security exit initialization data:

Initialization data to be passed to the security exit.

Data type String

Migrating a listener port to an activation specification for use with the WebSphere MQ messaging provider

For WebSphere Application Server Version 7 and later, listener ports are stabilized. You must therefore plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications.

Before you begin

EJB 2.0 message-driven beans cannot be configured against JCA 1.5-compliant resources. If your bean is an EJB 2.0 application, upgrade it to EJB 3 or EJB 2.1 before you complete this task.

Note: You can continue to configure EJB 3, EJB 2.1, and EJB 2.0 message-driven beans against a listener port. You might want to do this for compatability with existing message-driven bean applications. However, listener ports are stabilized, and you should plan to migrate all your message-driven beans to use JCA 1.5-compliant resources. For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

For WebSphere Application Server Version 7 and later, listener ports are stabilized. For more information, read the article on stabilized features. You should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications. However, you should not begin this migration until you are sure the application does not have to work on application servers earlier than WebSphere Application Server Version 7. For example, if you have an application server cluster with some members at Version 6.1 and some at Version 7, you should not migrate applications on that cluster to use activation specifications until after you migrate all the application servers in the cluster to Version 7.

Note that the **Maximum retries** listener port setting is not migrated to the new activation specification as there is no exact equivalent.

When you are migrating a listener port associated with a message-driven bean (MDB) that has the subscriptionDurability activation configuration property set to Durable, and that MDB already has an active durable subscription, the durable subscription is not migrated. This is because listener ports and WebSphere MQ activation specifications use incompatible forms of subscription name. As a result there can be two active durable subscriptions subscribed to the relevant topic for the same MDB. As part of the migration process, you must delete the old durable subscription that was associated with the listener port and manually clean up any messages associated with it. For information on how do to this see the WebSphere MQ and WebSphere Message Broker information centres.

Procedure

1. Start the administrative console.
2. In the navigation pane, expand**Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.
3. Select the listener port that you want to work with by selecting the check box to the left of the listener port name.
4. Click **Convert to activation specification** to start the "Convert listener port to activation specification" wizard.
5. On the "Step1: Supply activation specification name" page, enter the following information then click **Next** to continue:

- The name of the new activation specification to be created.
- The JNDI name of the new activation specification.
- The scope of the new activation specification (Server, Node, Cluster, Cell). Note that Cluster only appears when the server is in a cluster.

6. On the "Step2: Summary" page, click **Finish** to complete the creation of the new activation specification.
7. Stop then restart the application server.
8. To complete the configuration of the activation specification, see “Configuring an activation specification for the WebSphere MQ messaging provider” on page 738.

Creating a connection factory for the WebSphere MQ messaging provider

Use this task to create a connection factory, a queue connection factory, or a topic connection factory for use with the WebSphere MQ messaging provider.

About this task

With JMS 1.1, domain-independent connection factories are preferred to domain-specific queue connection factories and topic connection factories.

To create a connection factory, a queue connection factory, or a topic connection factory for use with the WebSphere MQ messaging provider, use the administrative console to complete the following steps. You can choose either to enter all the required connection information using the Create WebSphere MQ JMS resource wizard or to use a client channel definition table (CCDT) to establish a connection to the WebSphere MQ messaging provider.

For information about modifying an existing connection factory, see the related tasks.

Procedure

1. In the navigation pane, select the type of connection factory you want to create:
   - To create a connection factory, click **Resources** > **JMS->Connection factories**.
   - To create a queue connection factory, click **Resources** > **JMS->Queue connection factories**.
   - To create a topic connection factory, click **Resources** > **JMS->Topic connection factories**.
2. Select the **Scope** setting corresponding to the scope of the connection factory that you want to create.
3. Click **New** in the content pane to start the Create WebSphere MQ JMS resource wizard.
4. Select **WebSphere MQ messaging provider**, then click **OK**.
5. On the “Configure basic attributes” page, specify the following properties, then click **Next**.

   Name The name by which this connection factory is known for administrative purposes within WebSphere Application Server.

   JNDI name
   The name that is used to bind this connection factory into the Java Naming and Directory Interface (JNDI) namespace.

   Description
   Optional. A description of this connection factory for administrative purposes within WebSphere Application Server.
6. On the “Select connection method” page, choose how you want to specify the connection details for to the WebSphere MQ messaging provider by selecting one of the following options, then click **Next**.

Enter all the required information into this wizard
Choosing this option takes you to the “Supply queue manager details” page (see step 7), where you can start entering the connection details using the Create WebSphere MQ JMS resource wizard.

Use a client channel definition table
Choosing this option takes you to the “Specify client channel definition table” page (see step 8), where you can enter details of the client channel definition table that you want to use.

7. If you have chosen to enter all the required information using the wizard, complete the following steps:
   a. On the “Supply queue manager details” page, enter the name of the queue manager or queue-sharing group that you want to connect to then click **Next**.
   b. On the “Enter connection details” page, specify the details for the connection, then click **Next** to continue to the “Test connection” page. You can choose either to enter host and port information separately, or enter host and port information in the form of a connection name list. You must only choose the option to use a connection name list if you are creating a connection to a multi-instance queue manager. You must not use this option for connections to non-multi-instance queue managers as that can result in transaction integrity issues.

      Transport
      Optional. The WebSphere MQ transport type for the connection. The option that you select is used to determine the exact mechanisms used to connect to WebSphere MQ. If you are configuring a transport type of *bindings, then client* or *bindings*, see “Configuring the WebSphere MQ messaging provider with native libraries information” on page 723 for more information.

      Enter host and port information in the form of separate hostname and port values
      This radio button is selected by default and enables the **Hostname** and **Port** fields and disables the **Connection name list** field. Leave this radio button selected if you want to enter host and port information in the form of separate host name and port values.

      Hostname
      The host name, IPv4, or IPv6 address of the WebSphere MQ queue manager to connect to. Complete this field if you have selected **Enter host and port information in the form of separate hostname and port values**.

      Port Optional. The port number on which WebSphere MQ is listening. Complete this field if you have selected **Enter host and port information in the form of separate hostname and port values**.

      Enter host and port information in the form of a connection name list
      This radio button is cleared by default and, if selected, disables the **Hostname** and **Port** fields and enables the **Connection name list** field.
      - Click this radio button to select it if you want to enter host and port information in the form of a connection list.
      - Leave this radio button cleared if you want to enter host and port information in the form of separate hostname and port values.

      Connection name list
      The connection name list specifying the host name and port details to use when you want the connection factory to connect to a multi-instance queue manager. Complete this field if you have selected **Enter host and port information in the form of a connection name list**. Enter the host name and port details in the following form:

      `host[(port)][,host(port)]`

      host must be a valid TCP/IP host name or IPv4 or IPv6 address.

      port must be an integer value in the range 1 - 65536 (inclusive). The port information is optional, and if not specified, defaults to 1414.

For example: `localhost(1234),remotehost1(1234),remotehost2`

Server connection channel
Optional. The WebSphere MQ server connection channel name used when connecting to WebSphere MQ queue manager or queue-sharing group.

8. If you are using a channel client definition table to establish a connection to the WebSphere MQ messaging provider, complete the following fields on the “Specify client channel definition table” page, then click **Next** to continue to the “Test connection” page.

   Client channel definition table URL
   The URL to the client channel definition table that you want to use when connecting to WebSphere MQ.

   Queue manager
   Optional. The name of the queue manager to be used to select one or more entries from the CCDT.
9. Optional: On the “Test connection” page, if you want to test establishing the connection, click **Test connection**. This test can take several seconds to perform.
10. On the “Summary” page, complete the creation of the new connection factory by clicking **Finish**.
11. Stop then restart the application server.

JMS resource provider selection panel

Select the messaging provider with which to create this JMS activation specification, connection factory or destination.

You select the scope setting on an earlier page. The choice of JMS provider depends on the scope that you selected. You might see a choice such as the following list:

- Default messaging provider.

  Select this option if you want the JMS resource to be provided by the service integration bus, as part of WebSphere Application Server.
- WebSphere MQ messaging provider

  Select this option if you want the JMS resource to be provided by WebSphere MQ. You must have installed and configured a WebSphere MQ network in order to use this provider.
- “My JMS provider”

  Select this option if you want the JMS resource to be provided by a third-party JMS provider. This option is only available if you have installed and configured a third-party provider. The name that is displayed (for example“My JMS provider”) is the name you gave to the provider when you installed and configured it.
- V5 default messaging provider

  Select this option if you want to manage the JMS connection factories and destinations for JMS applications that were designed to work with WebSphere Application Server Version 5.1, and are still running on a Version 5.1 or Version 6 node that is part of a Version 7.0 or later mixed cell.

Configuring a unified connection factory for the WebSphere MQ messaging provider

Use this task to view or change the configuration of an existing domain-independent connection factory for the WebSphere MQ messaging provider on a WebSphere Application Server node.

About this task

This task applies to unified connection factories. With JMS 1.1, domain-independent (unified) connection factories are preferred to domain-specific queue connection factories and topic connection factories. If you want to view or change a queue connection factory or topic connection factory, see the related tasks.

To view or change the configuration of an existing connection factory for use with the WebSphere MQ messaging provider, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Resources** > **JMS->Connection factories**. A list of existing unified connection factories, with a summary of their properties, is displayed.
2. Select the **Scope** corresponding to the scope at which the connection factory is visible to applications.
3. Click the name of the connection factory that you want to view or change. Configuration details for the unified connection factory are displayed.
4. Under **General Properties** make any required changes.

   For information about each of the available fields, see "WebSphere MQ messaging provider connection factory settings."
5. Click **Apply** to save the configuration.
6. Optional: Click **Advanced properties** to display or change the list of advanced properties of your connection factory.
7. Optional: Click **Broker properties** to display or change the list of broker properties of your connection factory.
8. Optional: Click **Custom properties** to display or change the list of custom properties of your connection factory.
9. Optional: Click **Client transport properties** to display or change the list of client transport properties of your connection factory. This link is only appears if, when you created the connection factory, you chose to enter all the information required to connect to WebSphere MQ. This link does not appear if you have chosen to use a client channel definition table (CCDT) to establish a connection to WebSphere MQ.
10. Optional: Click **Connection pools** to display or change the connection pools detail of your connection factory.
11. Optional: Click **Session pools** to display or change the session pools detail of your connection factory.

    There is a mechanism (session.sharing.scope custom property) to make JMS sessions unshareable. This means that whenever an application calls Session.close(), the JMS session is automatically released from any transaction that is associated with and returned to the session pool. This means that sessions can be cleaned up and removed from the session pool even if the servlet or asynchronous bean that created it is still running.
12. Click **OK**.
13. Save your changes to the master configuration.
14. To make the changed configuration take effect, stop then restart the application server.

WebSphere MQ messaging provider connection factory settings

Use this panel to view or change the configuration properties of the selected connection factory for use with the WebSphere MQ messaging provider. These configuration properties control how connections are created to associated JMS queues and topics.

The WebSphere MQ messaging provider supports JMS 1.1 domain-independent interfaces, such as the unified JMS connection factory. A domain-independent application can use the same interface for both point-to-point and publish/subscribe messaging, and can support both point-to-point and publish/subscribe messaging within the same transaction. With JMS 1.1, you are recommended to use domain-independent unified JMS connection factories for new applications. A domain-specific interface extends the domain-independent equivalent, so an application that uses domain-specific queue and topic connection factories can choose to use either interface.

To view WebSphere MQ connection factory settings, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->Connection factories** to display existing connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the connection factories are defined. This restricts the set of connection factories displayed.
3. Select the name of the connection factory that you want to work with.

Under General Properties there are four groups of properties:

- "Administration" on page 764
- "Connection" on page 765
- "Advanced" on page 770
- "Security settings" on page 771

Make any required changes to the Administration, Connection, Advanced, and Security settings groups of properties, and then click **Apply** to save the configuration before, in the content pane under Additional Properties, you click any of the following links:

- **Advanced properties** to display or change the advanced properties of your WebSphere MQ connection factory.
- **Broker properties** to display or change the broker properties of your WebSphere MQ connection factory.
- **Custom properties** to display or change the custom properties of your WebSphere MQ connection factory.
- **Client transport properties** to display or change the client transport properties of your WebSphere MQ connection factory. If the selected connection factory was not created using a Client Channel Definition Table (CCDT), follow this link to enter all the information required to connect to WebSphere MQ. If the selected connection factory was created using a CCDT, you do not need to supply the client transport properties, and so the link is absent.
- **Connection pools** to display or change the connection pools detail of your WebSphere MQ connection factory.
- **Session pools** to display or change the session pools detail of your WebSphere MQ connection factory.

Under Related Items, you can click **JAAS - J2C authentication data** to configure authentication information for use with the connection factory.

You can specify the following additional properties by using WebSphere MQ administrative commands:

- **-localAddress**
- **-clonedSubs**
- **-componentAuthAlias**

For more information about these properties, refer to the "createWMQConnectionFactory command" on page 888.

Note: When specifying WebSphere MQ properties, the following restrictions apply:

- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ JMS resources, see the *Using Java* and *WebSphere MQ System Administration* sections of the WebSphere MQ information center.

If WebSphere MQ functionality has been disabled at a scope that affects this WebSphere MQ messaging provider resource, then an informational message indicating that WebSphere MQ has been disabled appears. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

A WebSphere MQ unified connection factory has the following properties.

Administration:

Scope:

The level at which this connection factory definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change other resources at a different scope, change the scope on the WebSphere MQ connection factory collection panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Provider:

The JMS provider assigned when the queue connection factory is created.

For all connection factories that use this panel, the provider is the WebSphere MQ messaging provider.

The provider is displayed for information only.

Data type String

Name:

The name by which this connection factory is known for administrative purposes within WebSphere Application Server.

Data type String
Range The name must be unique within the set of connection factories defined to the cell.

JNDI name:

The JNDI name that is used to bind the connection factory into the JNDI namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

Data type String

Description:

A description of this connection factory for administrative purposes within WebSphere Application Server.

Data type String
Default Null

Connection:

The information required to configure a connection depends on whether the selected queue connection factory was created using a Client Channel Definition Table (CCDT).

If the selected connection factory was created using a CCDT, only the following properties are displayed:
- Client channel definition table URL
- Queue manager
- SSL configuration

If the selected connection factory was not created using a CCDT, the following properties are displayed:
- Queue manager
- Transport
- If **Enter host and port information in the form of separate host and port values** is selected, the connection name list property cannot be used and the following properties can be used:
  - Host name
  - Port
- If **Enter host and port information in the form of a connection name list** is selected, the connection name list property can be used and the following properties cannot be used:
  - Host name
  - Port
- Server connection channel
- If you clear the check box for the **Use SSL to secure communication with Websphere MQ** property, the following properties cannot be used:
  - Centrally managed
  - Specific configuration
  - SSL configuration

For more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *Using Java* section of the WebSphere MQ information center.

Client channel definition table URL:

A URL that specifies the location of a WebSphere MQ CCDT.

Data type String

Queue manager:

If the specified connection factory is based on a CCDT, this property is used to select an entry in the CCDT. Otherwise, this property specifies the name of the queue manager or queue-sharing group to connect to. A connection is established to the specified WebSphere MQ resource to send or receive messages.

Data type String

Range

If this connection factory is not based on a CCDT, the value must be a valid queue manager name.

If this connection factory is based on a CCDT, the value must be one of the following:

- A valid queue manager name
- An asterisk (*) followed by the name of a queue manager group[1]
- An asterisk (*)
- Blank[1]

[1]When you specify the value of the Queue manager property in this form in combination with a CCDT, individual connections established by using the connection factory might connect to different queue managers. Selection from multiple queue managers occurs when the CCDT contains multiple client connection channel definitions with a matching queue manager name (QMNAME) parameter, and these connection channel definitions define the network connection details of different queue managers.

If the specified connection factory is based on a CCDT, and the CCDT can select from more than one queue manager, you might not be able to recover global transactions. Therefore, for connection factories that specify a CCDT, you have two alternatives:

- Avoid any ambiguity about the target queue manager when specifying the Queue manager property, which means that the specified value of this property must not include an asterisk (*).
- Avoid using the resources with applications that enlist in global transactions.

Transport:

The WebSphere MQ transport type for the connection. The transport type is used to determine the exact mechanisms used to connect to WebSphere MQ.

Data type Drop-down list
Default **Bindings, then client**

Range

Client Use a TCP/IP-based network connection to communicate with the WebSphere MQ queue manager.

Bindings, then client
Attempt a bindings mode connection to the queue manager. If this is not possible, revert to the client transport.

Bindings
Establish a cross-memory connection to a queue manager running on the same node. The following Client Transport Mode properties are disabled:
- Host name
- Port
- Connection name list
- Server connection channel

For more information about configuring a transport type of *Bindings, then client* or *Bindings*, refer to "Configuring the WebSphere MQ messaging provider with native libraries information" on page 723.

Enter host and port information in the form of separate host and port values:

If this radio button is selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the host name and port properties.

Selecting this option enables the host name and port properties, and disables the connection name list property. To enter connection name list information, click **Enter host and port information in the form of a connection name list**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type Radio button

Hostname:

The host name, IPv4, or IPv6 address of the WebSphere MQ queue manager to connect to.

Data type String

Port:

The port number on which WebSphere MQ is listening.

Data type Integer
Default 1414
Range The value must be in the range 1 to 65536 (inclusive).

Enter host and port information in the form of a connection name list:

If this radio button is selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the connection name list property.

Connection name lists can be used to connect to a single queue manager or to a multi-instance queue manager. For more information on using a multi-instance queue manager, see the WebSphere MQ information centre. Selecting this option enables the connection name list property and disables the host name and port properties. To enter separate host and port information, click **Enter host and port information in the form of separate host and port values**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type Radio button

Connection name list:

The connection name list specifying the host name and port details to use when you want the connection factory to connect to a multi-instance queue manager.

This property must only be used to allow connection to a multi-instance queue manager. It must not be used to allow connections to non-multi-instance queue managers as that can result in transaction integrity issues.

Data type String
Default Unset
Range This field must be set to a string in the following form:

host[(*port*)][,*host*(*port*)]

The port information is optional, and if not specified, defaults to 1414.

host must be a valid TCP/IP host name or IPv4 or IPv6 address.

port must be an integer value in the range 1 to 65536 (inclusive).

For example:
`localhost(1234),remotehost1(1234),remotehost2`

When the connection name list property is specified, the host name or port properties are automatically set to the host name and port number of the first entry in the connection name list. So if you specified `localhost(1234),remotehost1(1234),remotehost2`, the host name would be set to localhost and port would be set to 1234.

This property is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

When used in a mixed cell environment, the information in the connection name list property is not available to versions of WebSphere Application Server earlier than Version 8.0. In this case, the information in the host name and port name fields, based on the first element in the connection name list, is used instead.

Server connection channel:

The WebSphere MQ server connection channel name used when connecting to WebSphere MQ.

Data type String
Default SYSTEM.DEF.SVRCONN

Range

The value must be a server connection channel defined to the WebSphere MQ queue manager that is being connected to.

Use SSL to secure communications with WebSphere MQ:

This option determines whether the SSL (Secure Sockets Layer) protocol is used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

When using a WebSphere MQ messaging provider connection factory in the application server environment, the application server manages SSL configuration. To change SSL configuration parameters, use the administrative console to navigate to the **Security** > **SSL certificate and key management** panel.

When using a WebSphere MQ messaging provider connection factory in the client environment, the client takes SSL configuration information from the ssl.client.props file. Use of this file is detailed in the related reference information for this topic.

You can only use one cipher suite in the SSL configuration for a WebSphere MQ messaging provider connection factory. If you specify more than one cipher suite, only the first one is used.

Data type

Check box. If this check box is cleared, the following SSL properties are disabled:

- Centrally managed
- Specific configuration
- SSL configuration

Centrally managed:

When the SSL protocol is used to communicate with WebSphere MQ, select this radio button to specify that the SSL configuration is taken from the centrally managed WebSphere Application Server SSL configuration.

When you select this radio button, the host name and port attributes from the WebSphere MQ messaging provider connection factory are used to select an appropriate SSL configuration. If host and port information has been supplied by a connection name list this means that the host name and port information of the first element in the list are used to select an appropriate SSL configuration. To provide the SSL configuration which will be matched to the connection factory, see the Dynamic outbound endpoint SSL configuration settings topic listed under related reference.

Data type

Radio button

Specific configuration:

Select this radio button when you want to specify a particular SSL configuration for use when SSL is to be used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

Data type

Radio button

SSL configuration:

The SSL configuration to use when SSL is to be used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

This property is disabled if the **Centrally managed** radio button is selected and the WebSphere MQ messaging provider resource has been explicitly defined.

This property is always enabled if the WebSphere MQ messaging provider resource is based on a CCDT.

If this WebSphere MQ messaging provider resource is based on a CCDT, this parameter is only used if the relevant entries in the CCDT have been configured to use SSL.

Additionally, if an SSL configuration of `none` is selected, the default centrally managed WebSphere Application Server SSL configuration for the WebSphere MQ messaging provider is used.

Data type Drop-down list

Advanced:

Client ID:

The client identifier to specify when connecting to the WebSphere MQ messaging provider.

The client identity is not a user ID in the conventional sense, and is not related to security. It is used by durable subscriptions in publish-subscribe messaging. A durable subscription continues to collect up messages while the subscriber is "away" (for example, not running or failed) and delivers those messages when the subscriber reconnects. The client identity is the token that says which subscriber you are when you reconnect so that you get the messages that have been saved up for you in your absence.

Data type String

Allow cloned durable subscriptions:

An option that determines whether multiple instances of a durable subscription can be accessed concurrently by different servers.

Data type Check box
Default Cleared
Range
Selected
Multiple instances of a durable subscription can be accessed concurrently by different servers.
Cleared
Multiple instances of a durable subscription cannot be accessed concurrently by different servers.

Provider version:

The WebSphere MQ messaging provider version. This value is used to determine whether to connect to a particular version of a queue manager. It is also used to determine the type of functions required by the client.

Data type String

Range

The value entered must either be the empty string or be in one of the following formats:

n.n.n.n

n.n.n

n.n

n

where n is a numeric value greater than or equal to zero.

For example 6.0.0.0.

Support distributed two-phase commit protocol:

An option that specifies whether the connection factory supports XA coordination of messaging transactions. Enable this option if multiple resources, including this connection factory, are to be used in the same transaction.

If you clear this property, you disable support for distributed two-phase commit protocol. The JMS session can still be enlisted in a transaction, but it uses the resource manager local transaction calls session.commit and session.rollback, instead of XA calls. This can lead to an improvement in performance. However, only a single resource can be enlisted in a transaction in WebSphere Application Server.

Last participant support enables you to enlist one non-XA resource with other XA-capable resources.

Data type Check box

Default Selected

Range

Selected
The connection factory supports the use of distributed two-phase commit protocols for the coordination of transacted work.

Cleared
The connection factory does not support the use of distributed two-phase commit protocols for the coordination of transacted work.

Keep this option selected if transactions involve other resources, including other queues or topics. Clear this option only when you are certain that the queue manager that is the target for this queue connection is the only resource in the transaction.

Security settings:

Authentication alias for XA recovery:

The user name and password to use when connecting to WebSphere MQ during XA recovery.

Data type Drop-down list

Default (none)

Range All authentication aliases defined to the cell and the value "(none)", which specifies that no credentials are passed to WebSphere MQ during XA recovery.

Mapping-configuration alias:

This field is used only in the absence of a login configuration on the component resource reference.

When the resource authority value is "container", the preferred way to define the authentication strategy is by specifying a login configuration and associated properties on the component resource reference.

If the **DefaultPrincipalMapping** login configuration is specified, the associated property is a JAAS - J2C authentication data entry alias. To configure authentication information for use with the connection factory, under Related Items, click **JAAS - J2C authentication data** .

Data type	Drop-down list
Default	(none)
Range	The following options are available: ClientContainer WSLogin WSKRB5Login DefaultPrincipalMapping TrustedConnectionMapping KerberosMapping

Container-managed authentication alias:

The authentication alias which specifies the user name and password to use when connecting to the WebSphere MQ messaging provider.

Data type	Drop-down list
Default	(none)
Range	All authentication aliases defined to the cell and the value "(none)", which specifies that no credentials are passed to WebSphere MQ.

WebSphere MQ messaging provider connection factory advanced properties:

Use this panel to view or change the advanced properties of the selected connection factory for use with the WebSphere MQ messaging provider. These advanced properties control the behavior of connections made to WebSphere MQ messaging provider destinations.

To view WebSphere MQ connection factory advanced properties, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->Connection factories** to display existing connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the connection factories are defined. Setting the scope in this way restricts the set of queue connection factories displayed.
3. Select the name of the connection factory that you want to work with.
4. In the content pane, under Additional properties, click **Advanced properties** to view a list of the advanced properties of the WebSphere MQ connection factory.

Under General Properties there are six groups of advanced properties:

- Client reconnect
- Message compression
- Temporary destinations
- Connection consumer
- Message format
- Additional

Make any required changes to these groups and then click **Apply** to return to the connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, except for channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring JMS resources for WebSphere MQ. For more information about configuring JMS resources for WebSphere MQ, see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ connection factory has the following advanced properties:

Client reconnect options:

This property specifies whether a client mode connection reconnects automatically, or not, in the event of a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment.

Data type Drop-down list
Default DISABLED
Range
DISABLED Client reconnection does not automatically occur.
ASDEF The value from the DefRecon attribute from the channels stanza of the client configuration file is used. If there is no DefRecon value specified in the client configuration file, ASDEF has the same effect as DISABLED.
QMGR Reconnection occurs only to the queue manager to which the connection was originally connected.
RECONNECT Reconnection occurs to any queue manager that is consistent with the value of the queue manager attribute. This queue manager might be a different queue manager from the one to which the connection was originally connected.

For more information about automatic client reconnection, see the WebSphere MQ information center.

Client reconnect timeout:

The maximum number of seconds that a client mode connection spends attempting to automatically reconnect to a queue manager after a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment. Whether this parameter is used or not depends on the value of the client reconnect options parameter.

Data type Integer
Units Seconds
Default 1800 (30 minutes)
Range A value greater than zero and up to 2147483647

For more information about automatic client reconnection, see the WebSphere MQ information center.

Compress message headers:

An option that enables the compression of message headers.

Data type Check box

Default Cleared

Range

Cleared
Do not compress message headers.

Selected
Compress message headers.

Compression algorithm for message payloads:

The compression algorithm that is used to compress message payloads.

Data type Drop-down list

Default NONE

Range

RLE Message data compression is performed using run-length encoding.

ZLIBFAST
Message data compression is performed using ZLIB encoding with speed prioritized.

ZLIBHIGH
Message data compression is performed using ZLIB encoding with compression prioritized.

NONE No message data compression is performed.

WebSphere MQ model queue name:

The model queue that is used as a basis for temporary queue creation.

Data type String

Default SYSTEM.DEFAULT.MODEL.QUEUE

Temporary queue prefix:

The prefix that is appended to the beginning of the names generated for temporary queues.

Data type String

Temporary topic prefix:

The prefix that is appended to the beginning of the names generated for temporary topics.

Data type String

Retain messages, even if no matching consumer is available:

An option that determines whether messages for which there is no matching consumer are retained on the input queue or dealt with according with their disposition options.

Data type Check box

Default Selected

Range

Cleared
Do not retain messages.

Selected
Retain messages.

Polling interval:

This setting is applicable in the client container only. When using aWebSphere MQ Version 6 queue manager (or WebSphere MQ Version 5.3 for z/OS), this setting configures the mechanism used to dispatch messages to JMS asynchronous consumers.

This setting is used when the set of WebSphere MQ queues that is being asynchronously consumed from exceeds the number of threads that are available internally to synchronously get messages from the WebSphere MQ queue. The setting determines how long a thread waits for a message to arrive at a WebSphere MQ queue before polling another WebSphere MQ queue in the set.

Data type	Integer
Units	Milliseconds
Default	5000
Range	A value greater than zero.

Rescan interval:

When using a WebSphere MQ Version 6 queue manager (or WebSphere MQ Version 5.3 for z/OS), this setting configures the mechanism used to dispatch messages to JMS asynchronous consumers.

This setting is used when the set of WebSphere MQ queues that is being asynchronously consumed from exceeds the number of threads that are available internally to synchronously get messages from the WebSphere MQ queue. The setting determines how long a thread retrieves messages from a WebSphere MQ queue before switching to use messages from another WebSphere MQ queue in the set.

Data type	Integer
Units	Milliseconds
Default	5000
Range	A value greater than zero.

Maximum batch size:

The maximum number of messages that can be removed from a queue before at least one must be delivered to an asynchronous consumer.

Data type	Integer
Default	10
Range	A value greater than zero.

Coded character set identifier:

The character set to use when you are encoding strings in the message.

Data type	Integer
Default	819
Range	A value greater than zero. The coded character set identifier (CCSID) must be one of the CCSIDs supported by WebSphere MQ.

For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *System Administration* and *Application Programming Reference* sections of the WebSphere MQ information center.

Append an RFH version 2 header to reply messages:

The action required when sending a reply message to the reply-to queue obtained from a message that does not include an RFH version 2 header.

Data type Check box
Default Cleared
Range

Selected
Append an RFH version 2 header to reply messages regardless of whether the original message had an RFH version 2 header.

Cleared
Append an RFH version 2 header to reply messages only if the original message had an RFH version 2 header.

Fail JMS method calls if the WebSphere MQ queue manager is quiescing:

An option that enables selected JMS operations to fail when the queue manager is put into a quiescing state. Selecting this option enables the queue manager to quiesce successfully and shut down.

Data type Check box
Default Selected
Range

Cleared
Do not fail JMS operations if the queue manager is quiescing.

Selected
Fail JMS operations if the queue manager is quiescing.

WebSphere MQ messaging provider connection factory broker properties:

Use this panel to view or change the broker settings of the selected connection factory, or topic connection factory, for use with the WebSphere MQ messaging provider. These broker settings determine how the WebSphere MQ messaging provider interacts with a broker for the purposes of publishing messages and subscribing to topics. Updates to the settings take effect when the server is restarted.

To view WebSphere MQ connection factory, or topic connection factory, broker properties use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Connection factories** to display existing connection factories, or click **Resources** > **JMS->Topic connection factories** to display existing topic connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the connection factories are defined. This restricts the set of connection factories displayed.
3. Select the name of the connection factory, or topic connection factory, that you want to work with.
4. In the content pane, under Additional Properties, click **Broker properties** to display the broker properties of the WebSphere MQ connection factory, or topic connection factory.

Under General Properties there are four groups of properties:
- Queues
- Capabilities
- Tuning
- Additional

Make any required changes to these groups and then click **Apply** to return to the connection factory, or topic connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:

- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the WebSphere MQ *Using Java* section of the WebSphere MQ library.

A WebSphere MQ connection factory, or topic connection factory, has the following broker properties:

Broker control queue:

The queue to which broker control messages are sent.

Data type String
Default SYSTEM.BROKER.CONTROL.QUEUE

Broker publication queue:

The queue to which publication messages are sent.

Data type String
Default SYSTEM.BROKER.DEFAULT.STREAM

Broker subscriber queue:

The queue to which subscription messages are sent.

Data type String
Default SYSTEM.JMS.ND.SUBSCRIBER.QUEUE

Broker connection consumer subscription queue:

The queue to which subscription messages that are destined for a connection consumer are sent.

Data type String
Default SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE

Version:

The version of the broker that is used. This determines some of the capabilities that the broker is assumed to have. For example, whether to use an RFH version 1 or version 2 header in publications.

Data type Radio button
Default Version 1 broker
Range

Version 1 broker
Message selection cannot be specified.

Version 2 broker
Message selection can be specified. If you select this option, you must also complete **Specify where message selection occurs**.

Specify where message selection occurs:

The process in which message selection is performed. This property is enabled only if **Version 2 broker** was selected.

Data type Drop-down list
Default CLIENT
Range

CLIENT
Message selection is performed in the application server process.

BROKER
Message selection is performed in the broker process.

Subscription store:

The process for tracking subscriptions.

Data type Drop-down list
Default MIGRATE
Range

MIGRATE
Any information that is held on queues is migrated to the broker mechanism for persisting subscription information. If subscription information is already persisted using the broker mechanism then specifying a value of `Migrate` is equivalent to specifying a value of `Broker`.

BROKER
Internal broker mechanisms are used to track subscription information.

QUEUE
A designated WebSphere MQ queue is used to record information about current subscriptions.

Durable subscription state refresh interval:

How often a long running transaction is recreated and used to clean up durable subscriptions, for some versions of the queue manager.

Data type Integer
Default 60000
Range Any positive integer

Subscription cleanup level:

How aggressively messages are cleaned up if the subscriber that is expected to consume the messages terminates unexpectedly.

Data type Drop-down list
Default SAFE

Range

SAFE A conservative algorithm is used to clean up subscriptions.

ASPROP
The cleanup algorithm is determined by a system property.

NONE No cleanup of subscriptions is performed.

STRONG
An aggressive algorithm is used to clean up subscriptions.

Subscription cleanup interval:

How often to check for orphaned subscriptions and clean up messages.

Data type Integer
Default 3600000
Range Any positive integer

Subscription wildcard format:

The wildcard format used for subscribing to more than one topic in a topic hierarchy.

Data type Drop-down list
Default character wildcards
Range

character wildcards
You can use an asterisk (*) or question mark (?) to represent characters, or strings of characters, in a topic name.

* is interpreted as matching many characters.

? is interpreted as matching a single character.

topic level wildcards
You can use a plus sign (+) or number sign (#) to represent topics in a multilevel topic hierarchy.

+ is interpreted as matching a single topic name.

is interpreted as matching many topics in the hierarchy. / is used to delimit topics.

Publish acknowledgement window:

The number of messages to publish before publishing a message that requires broker acknowledgement

Data type Integer
Default 25
Range Any positive integer

Optimize for sparse subscription patterns:

An option to specify whether this connection factory is anticipated to receive a high proportion of messages that match its selection criteria. This information can be used to optimize message delivery.

Data type Check box
Default Cleared
Range
Cleared
Subscriptions frequently receive matching messages.
Selected
Subscriptions do not frequently receive matching messages.

Broker queue manager:

The name of the queue manager that is running the broker, if it is not the same as the queue manager to which the connection factory connects.

Data type String
Default The queue manager name that was specified in the connection factory.

WebSphere MQ messaging provider connection factory client transport settings:

Use this panel to view or change the client transport settings of a connection factory, queue connection factory, or topic connection factory for use with the WebSphere MQ messaging provider. Client transport properties affect how a client connection is established with a WebSphere MQ queue manager or queue-sharing group. Updates to the settings take effect when the server is restarted.

To view WebSphere MQ connection factory, queue connection factory, or topic connection factory client transport settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS**.
2. Click **Connection factories**, **Queue connection factories**, or **Topic connection factories** to display existing connection factories, queue connection factories or topic connection factories.
3. Select the name of the connection factory, queue connection factory, or topic connection factory that you want to work with.
4. In the content pane under Additional Properties, click **Client transport properties** to view a list of the client transport settings of the WebSphere MQ connection factory, queue connection factory, or topic connection factory.

Under General Properties there are two groups of properties:

- Additional SSL settings (for more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *Using Java* section of the WebSphere MQ information center)
- Channel exits

Make any required changes to these groups and then click **Apply** to return to the connection factory, queue connection factory or topic connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ connection factory, queue connection factory, or topic connection factory has the following client transport settings properties:

Certificate revocation list:

A list of LDAP URLs pointing to LDAP repositories of SSL certificates that might have been revoked.

Data type	String
Default	No certificate revocation list
Range	The value must be a space-separated list of LDAP URLs.

Peer name:

A name (possibly including wildcards) that must match the distinguished name of the peer SSL certificate for a connection to be established.

Data type	String
Default	Do not check the distinguished name of the peer certificate.
Range	Validated according to the rules for a WebSphere MQ SSLPEER channel parameter.

Reset count:

The total number of bytes to transfer over an SSL connection before renegotiating the symmetric encryption keys used to secure the connection.

Data type	Integer
Default	0 (do not renegotiate)
Range	The value must be in the range 0 through 999,999,999 (inclusive).

Receive exits:

A comma-separated list of Java class names corresponding to receive exits to be loaded.

Data type	String

Receive exit initialization data:

Initialization data to be passed to the receive exit.

Data type	String

Send exits:

A comma-separated list of Java class names corresponding to send exits to be loaded.

Data type	String

Send exit initialization data:

Initialization data to be passed to the send exit.

Data type	String

Security exit:

A Java class name corresponding to the security exit to be loaded.

Data type String

Security exit initialization data:

Initialization data to be passed to the security exit.

Data type String

Connection pool settings:

Use this page to configure connection pool settings.

This administrative console page is common to JDBC data sources and JMS connection factories (unified, queue, or topic connection factories). To view this page, the path depends on the type of resource, but generally you select an instance of the resource type then click **Connection Pool**. For example:

- Click **Resources** > **JDBC** > **Data Sources** > ***data_source*** > **[Additional Properties] Connection pool properties**
- Click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Connection pool**

Note: Connection pooling is not supported in an application client. The application client calls the database directly and does not go through a data source. If you want to use the getConnection() request from the application client, configure the JDBC provider in the application client deployment descriptors, using Rational Application Developer or an assembly tool. The connection is established between application client and the database. Application clients do not have a connection pool, but you can configure JDBC provider settings in the client deployment descriptors.

Connection timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

This value indicates the number of seconds that a connection request waits when there are no connections available in the free pool and no new connections can be created. This usually occurs because the maximum value of connections in the particular connection pool has been reached.

For example, if Connection timeout is set to 300, and the maximum number of connections are all in use, the pool manager waits for 300 seconds for a physical connection to become available. If a physical connection is not available within this time, the pool manager initiates a `ConnectionWaitTimeout` exception. In most cases, you should not retry the getConnection() method; if a longer wait time is required you should increase the Connection timeout setting value. If a `ConnectionWaitTimeout` exception is caught by the application, review the expected connection pool usage of the application and tune the connection pool and database accordingly.

If the Connection timeout is set to 0, the pool manager waits as long as necessary until a connection becomes available. This happens when the application completes a transaction and returns a connection to the pool, or when the number of connections falls below the value of Maximum Connections, and a new physical connection is created.

If Maximum Connections is set to 0, an infinite number of physical connections are enabled, and the Connection timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Maximum connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. When this number is reached, no new physical connections are created. The requester waits until a physical connection that is currently in use returns to the pool, or until a ConnectionWaitTimeoutException error displays. For example, if the Max Connections value is set to 5, and there are 5 physical connections in use, the pool manager waits for the amount of time specified in Connection timeout for a physical connection to become free.

Knowing the number of connection pools that can potentially request connections from the backend, such as a DB2 database or a CICS server, helps you determine a value for the Maximum Connections property.

For multiple stand-alone application servers that use the same data source configuration, or J2C connection factory configuration, a separate physical connection pool exists for each server. If you clone these same application servers, WebSphere Application Server (base) implements a separate connection pool for each clone.

All of these connection pools correspond to the same data source or connection factory configuration. Therefore all of these connection pools can potentially request connections from the same backend resource, at the same time. The single Maximum Connections value that you set on this console panel applies to every one of these connection pools. Consequently, setting a high Maximum Connections value can result in a load of connection requests that overwhelms your backend resource.

Data type	Integer
Default	10
Range	0 to maximum integer If Max Connections is set to 0, the Connection timeout value is ignored.

Tip: For better performance, set the value for the connection pool lower than the value for the maximum thread pool connections of the web container. To configure this setting click **Servers** > **Server types** > **WebSphere application servers** > ***server*** > **Thread Pools**, and modify the web container property. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the processor load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Minimum connections:

Specifies the minimum number of physical connections to maintain.

If the size of the connection pool is at or below the minimum connection pool size, the Unused timeout thread does not discard physical connections. However, the pool does not create connections solely to ensure that the minimum connection pool size is maintained. Also, if you set a value for Aged timeout, connections with an expired age are discarded, regardless of the minimum pool size setting.

For example, if the Minimum Connections value is set to 3, and one physical connection is created, the Unused timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the Minimum Connections setting.

Data type	Integer
Default	1
Range	0 to max int

Reap time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if Reap Time is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the Unused timeout and Aged timeout settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused timeout and Aged timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused timeout, until it reaches the number of connections specified in Minimum Connections. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread, set Reap Time to `0`, or set both Unused timeout and Aged timeout to `0`. The recommended way to disable the pool maintenance thread is to set Reap Time to `0`, and Unused timeout and Aged timeout are ignored. However, if Unused timeout and Aged timeout are set to `0`, the pool maintenance thread runs, but only physical connections that timeout, because of non-zero timeout values, are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused timeout value higher than the Reap timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections exceeds the Minimum Connections setting. For example, if the unused timeout value is set to `120`, and the pool maintenance thread is enabled (Reap Time is not `0`), any physical connection that remains unused for 2 minutes is discarded.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting Aged timeout to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged timeout value higher than the Reap timeout value for optimal performance.

For example, if the Aged timeout value is set to `1200`, and the Reap Time value is not `0`, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. The only exception is if the connection is involved in a transaction when the aged timeout is reached, the application server will not discard the connection until after the transaction is completed and the connection is closed.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**.

Data type	String
Defaults	• EntirePool for J2C connection factories and JMS-related connection factories • EntirePool for WebSphere Version 4.0 data sources • EntirePool for current version data sources that you create through the administrative console • EntirePool for current version data sources that you script through wsadmin AdminConfig commands, starting JDBC templates that are built into WebSphere Application Server. For information about the command createUsingTemplate, see the topic, Commands for the AdminConfig object. • FailingConnectionOnly for data sources that you script in wsadmin tool without starting JDBC templates :

Range

EntirePool
All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and issues a stale connection Exception during the next operation on that connection. Subsequent getConnection() requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this closure is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice.

FailingConnectionOnly
Only the connection that caused the stale connection exception is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a possibility that the next getConnection() request from the application can return a connection from the pool that is also stale. The result is more stale connection exceptions.

The connection pretest function attempts to insulate an application from pooled connections that are not valid. When a backend resource, such as a database, goes down, pooled connections that are not valid might exist in the free pool. This is especially true when the purge policy is failingConnectionOnly; in this case, the failing connection is removed from the pool. Depending on the failure, the remaining connections in the pool might not be valid.

Session pool settings:

Use this page to configure session pool settings.

This administrative console page is common to JMS unified connection factories, queue connection factories and topic connection factories. To view this page, you select an instance of the resource type then click **Session pools**. For example, click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Session pools**.

Connection Timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

The wait is necessary when the maximum value of connections (**Max Connections**) to a particular connection pool is reached . For example, if *Connection Timeout* is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is *not* available within this time, the Pool Manager throws a ConnectionWaitTimeoutException. It usually does not make sense to retry the getConnection() method, because if a longer wait time is required, you should set the **Connection Timeout** setting to a higher

value. Therefore, if this exception is caught by the application, the administrator should review the expected usage of the application and tune the connection pool and the database accordingly.

If Connection Timeout is set to 0, the Pool Manager waits as long as necessary until a connection is allocated (which happens when the number of connections falls below the value of **Max Connections**).

If Max Connections is set to 0, which enables an infinite number of physical connections, then the Connection Timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Max Connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use returns to the pool, or a ConnectionWaitTimeoutException is thrown.

For example, if the Max Connections value is set to 5, and there are five physical connections in use, the pool manager waits for the amount of time specified in Connection Timeout for a physical connection to become free.

If Max Connections is set to 0, the Connection Timeout value is ignored.

For better performance, set the value for the connection pool lower than the value for the Max Connections option in the web container. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

If clones are used, one data pool exists for each clone. Knowing the number of data pools is important when configuring the database maximum connections.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the CPU load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Data type	Integer
Default	10
Range	0 to max int

Min Connections:

Specifies the minimum number of physical connections to maintain.

Until this number is reached, the pool maintenance thread does not discard physical connections. However, no attempt is made to bring the number of connections up to this number. If you set a value for Aged Timeout, the minimum is not maintained. All connections with an expired age are discarded.

For example if the **Min Connections** value is set to 3, and one physical connection is created, the Unused Timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the **Min Connections** setting.

Data type	Integer
Default	1
Range	0 to max int

Reap Time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if **Reap Time** is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in **Min Connections**. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread set Reap Time to 0, or set both Unused Timeout and Aged Timeout to 0. The recommended way to disable the pool maintenance thread is to set Reap Time to 0, in which case Unused Timeout and Aged Timeout are ignored. However, if Unused Timeout and Aged Timeout are set to 0, the pool maintenance thread runs, but only physical connections that timeout due to non-zero timeout values are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused Timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the **Min Connections** setting. For example, if the unused timeout value is set to 120, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the **Reap Time** value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged Timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting **Aged Timeout** to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged Timeout value higher than the **Reap Timeout** value for optimal performance. For example, if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that

remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, are affected by the Reap Time value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge Policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**. Java EE Connector Architecture (JCA) data sources can have either option. WebSphere Version 4.0 data sources always have a purge policy of **EntirePool**.

Data type	String
Default	FailingConnectionOnly
Range	**EntirePool** All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and throws a *StaleConnectionException* during the next operation on that connection. Subsequent *getConnection* requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice. **FailingConnectionOnly** Only the connection that caused the *StaleConnectionException* is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a good possibility that the next *getConnection* request from the application can return a connection from the pool that is also stale, resulting in more stale connection exceptions.

Configuring a queue connection factory for the WebSphere MQ messaging provider

Use this task to view or change the configuration of an existing JMS queue connection factory for point-to-point messaging with the WebSphere MQ messaging provider.

About this task

This task applies to queue connection factories. With JMS 1.1, domain-independent (unified) connection factories are preferred to domain-specific queue connection factories and topic connection factories. To configure a unified connection factory, see “Configuring a unified connection factory for the WebSphere MQ messaging provider” on page 761.

To view or change the configuration of an existing queue connection factory for use with the WebSphere MQ messaging provider, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Resources** > **JMS->Queue connection factories**. A list of existing queue connection factories, with a summary of their properties, is displayed.
2. Select the **Scope** setting corresponding to the scope at which the queue connection factory is visible to applications.
3. Click the name of the queue connection factory that you want to view or change. Configuration details for the queue connection factory are displayed.
4. Under **General Properties** make any required changes.

   For information about each of the available fields, see “WebSphere MQ messaging provider queue connection factory settings.”
5. Click **Apply** to save the configuration.
6. Optional: Click **Advanced properties** to display or change the list of advanced properties of your queue connection factory.
7. Optional: Click **Custom properties** to display or change the list of custom properties of your queue connection factory.
8. Optional: Click **Client transport properties** to display or change the list of client transport properties of your queue connection factory. This link is only appears if, when you created the connection factory, you chose to enter all the information required to connect to WebSphere MQ. This link does not appear if you have chosen to use a client channel definition table (CCDT) to establish a connection to WebSphere MQ.
9. Optional: Click **Connection pools** to display or change the connection pools detail of your queue connection factory.
10. Optional: Click **Session pools** to display or change the session pools detail of your queue connection factory.

    There is a mechanism (session.sharing.scope custom property) to make JMS sessions unshareable. This means that whenever an application calls Session.close(), the JMS session is automatically released from any transaction that is associated with and returned to the session pool. This means that sessions can be cleaned up and removed from the session pool even if the servlet or asynchronous bean that created it is still running.
11. Click **OK**.
12. Save your changes to the master configuration.
13. To have the changed configuration take effect, stop then restart the application server.

WebSphere MQ messaging provider queue connection factory settings

Use this panel to view or change the configuration properties of the selected queue connection factory for use with the WebSphere MQ messaging provider. These configuration properties control how connections are created to associated JMS queue destinations.

A WebSphere MQ queue connection factory is used to create JMS connections to queues provided by WebSphere MQ for point-to-point messaging.

To view WebSphere MQ queue connection factory settings, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->Queue connection factories** to display existing queue connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the queue connection factories are defined. This restricts the set of queue connection factories displayed.
3. Select the name of the queue connection factory that you want to work with.

Under General Properties there are four groups of properties:

- "Administration"
- "Connection" on page 792
- "Advanced" on page 798
- "Security settings" on page 799

Make any required changes to the Administration, Connection, Advanced, and Security settings groups of properties, and then click **Apply** to save the configuration before, in the content pane under Additional Properties, you select any of the following links:

- **Advanced properties** to display the advanced properties of your WebSphere MQ queue connection factory.
- **Custom properties** to display the custom properties of your WebSphere MQ queue connection factory.
- **Client transport properties** to display or change the client transport properties of your WebSphere MQ queue connection factory. If the selected queue connection factory was not created using a Client Channel Definition Table (CCDT), follow this link to enter all the information required to connect to WebSphere MQ. If the selected queue connection factory was created using a CCDT, you do not need to supply the client transport properties, and so the link is absent.
- **Connection pools** to display the connection pool detail of your WebSphere MQ queue connection factory.
- **Session pools** to display the session pools detail of your WebSphere MQ queue connection factory.

Under Related Items, you can click **JAAS - J2C authentication data** to configure authentication information for use with the queue connection factory.

You can specify the following additional properties by using WebSphere MQ administrative commands:

- **-localAddress**
- **-componentAuthAlias**

For more information about these properties, refer to the "createWMQConnectionFactory command" on page 888.

Note: When specifying WebSphere MQ properties, the following restrictions apply:

- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ JMS resources, see the *Using Java* and *System Administration* sections of the WebSphere MQ information center.

A WebSphere MQ queue connection factory has the following properties.

Administration:

Scope:

The level at which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the WebSphere MQ queue connection factory collection panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Provider:

The JMS provider assigned when the queue connection factory is created.

For all queue connection factories created using this panel, the provider is the WebSphere MQ messaging provider.

The provider is displayed for information only.

Data type String

Name:

The name by which this queue connection factory is known for administrative purposes within WebSphere Application Server.

Data type String
Range The name must be unique within the set of queue connection factories defined to the cell.

JNDI name:

The JNDI name that is used to bind the queue connection factory into the namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

Data type String

Description:

A description of this queue connection factory for administrative purposes within WebSphere Application Server.

Data type String
Default Null

Connection:

The information required to configure a connection depends on whether the selected queue connection factory was created using a Client Channel Definition Table (CCDT).

If the selected queue connection factory was created using a CCDT, only the following properties are displayed:

- Client channel definition table URL
- Queue manager
- SSL configuration

If the selected queue connection factory was not created using a CCDT, the following properties are displayed:

- Queue manager
- Transport
- If **Enter host and port information in the form of separate host and port values** is selected, the connection name list property cannot be used and the following properties can be used:
  - Host name
  - Port
- If **Enter host and port information in the form of a connection name list** is selected, the connection name list property can be used and the following properties cannot be used:
  - Host name
  - Port
- Host name
- Port
- Server connection channel
- If you clear the check box for the **Use SSL to secure communication with Websphere MQ** property, the following properties cannot be used:
  - Centrally managed
  - Specific configuration
  - SSL configuration

For more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *Using Java* section of the WebSphere MQ information center.

Client channel definition table URL:

A URL that specifies the location of a WebSphere MQ CCDT.

Data type String

Queue manager:

If the specified queue connection factory is based on a CCDT, this property is used to select an entry in the CCDT. Otherwise, this property specifies the name of the queue manager or queue-sharing group to connect to. A connection is established to the specified WebSphere MQ resource to send or receive messages.

Data type String

Range

If this queue connection factory is not based on a CCDT, the value must be a valid queue manager name.

If this queue connection factory is based on a CCDT, the value must be one of the following:

- A valid queue manager name
- An asterisk (*) followed by the name of a queue manager group[1]
- An asterisk (*)
- Blank[1]

[1]When you specify the value of the Queue manager property in this form in combination with a CCDT, individual connections established by using the queue connection factory might connect to different queue managers. Selection from multiple queue managers occurs when the CCDT contains multiple client connection channel definitions with a matching queue manager name (QMNAME) parameter, and these connection channel definitions define the network connection details of different queue managers.

If the specified connection factory is based on a CCDT, and the CCDT can select from more than one queue manager, you might not be able to recover global transactions. Therefore, for connection factories that specify a CCDT, you have two alternatives:

- Avoid any ambiguity about the target queue manager when specifying the Queue manager property, which means that the specified value of this property must not include an asterisk (*).
- Avoid using the resources with applications that enlist in global transactions.

Transport:

The WebSphere MQ transport type for the connection. The transport type is used to determine the exact mechanisms used to connect to WebSphere MQ.

Data type Drop-down list
Default **Bindings, then client**

Range

Client Use a TCP/IP-based network connection to communicate with the WebSphere MQ queue manager.

Bindings, then client
Attempt a bindings mode connection to the queue manager. If this is not possible, revert to the client transport.

Bindings
Establish a cross-memory connection to a queue manager running on the same node. The following Client Transport Mode properties are disabled:
- Hostname
- Port
- Connection name list
- Server connection channel

For more information about configuring a transport type of *Bindings, then client* or *Bindings*, refer to "Configuring the WebSphere MQ messaging provider with native libraries information" on page 723.

Enter host and port information in the form of separate host and port values:

If this radio is button selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the host name and port properties.

Selecting this option enables the host name and port properties, and disables the connection name list property. To enter connection name list information, click **Enter host and port information in the form of a connection name list**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type Radio button

Hostname:

The host name, IPv4, or IPv6 address of the WebSphere MQ queue manager to connect to.

Data type String

Port:

The port number on which WebSphere MQ is listening.

Data type Integer
Default 1414
Range The value must be in the range 1 to 65536 (inclusive).

Enter host and port information in the form of a connection name list:

If this radio button is selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the connection name list property.

Connection name lists can be used to connect to a single queue manager or to a multi-instance queue manager. For more information on using a multi-instance queue manager, see the WebSphere MQ information centre. Selecting this option enables the connection name list property and disables the host name and port properties. To enter separate host and port information, click **Enter host and port information in the form of separate host and port values**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type	Radio button

Connection name list:

The connection name list specifying the host name and port details to use when you want the connection factory to connect to a multi-instance queue manager.

This property must only be used to allow connection to a multi-instance queue manager. It must not be used to allow connections to non-multi-instance queue managers as that can result in transaction integrity issues.

Data type	String
Default	Unset
Range	This field must be set to a string in the following form: *host*[(*port*)][,*host*(*port*)] The port information is optional, and if not specified, defaults to 1414. *host* must be a valid TCP/IP host name or IPv4 or IPv6 address. *port* must be an integer value in the range 1 to 65536 (inclusive). For example: `localhost(1234),remotehost1(1234),remotehost2`

When the connection name list property is specified, the host name or port properties are automatically set to the host name and port number of the first entry in the connection name list. So if you specified `localhost(1234),remotehost1(1234),remotehost2`, the host name would be set to localhost and port would be set to 1234.

This property is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

When used in a mixed cell environment, the information in the connection name list property is not available to versions of WebSphere Application Server earlier than Version 8.0. In this case, the information in the host name and port name fields, based on the first element in the connection name list, is used instead.

Server connection channel:

The WebSphere MQ server connection channel name used when connecting to WebSphere MQ.

Data type	String
Default	SYSTEM.DEF.SVRCONN

Range

The value must be a server connection channel defined to the WebSphere MQ queue manager that is being connected to.

Use SSL to secure communications with WebSphere MQ:

This option determines whether the SSL (Secure Sockets Layer) protocol is used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

When using a WebSphere MQ messaging provider queue connection factory in the application server environment, the application server manages SSL configuration. To change SSL configuration parameters, use the administrative console to navigate to the **Security** > **SSL certificate and key management** panel.

When using a WebSphere MQ messaging provider queue connection factory in the client environment, the client takes SSL configuration information from the ssl.client.props file. Use of this file is detailed in the related reference information for this topic.

You can only use one cipher suite in the SSL configuration for a WebSphere MQ messaging provider queue connection factory. If you specify more than one cipher suite, only the first one is used.

Data type

Check box. If this check box is cleared, the following SSL properties are disabled:

- Centrally managed
- Specific configuration
- SSL configuration

Centrally managed:

When the SSL protocol is used to communicate with WebSphere MQ, select this radio button to specify that the SSL configuration is taken from the centrally managed WebSphere Application Server SSL configuration.

When you select this radio button, the host name and port attributes from the WebSphere MQ messaging provider connection factory are used to select an appropriate SSL configuration. If host and port information has been supplied by a connection name list this means that the host name and port information of the first element in the list are used to select an appropriate SSL configuration. To provide the SSL configuration which will be matched to the connection factory, see the Dynamic outbound endpoint SSL configuration settings topic listed under related reference.

Data type

Radio button

Specific configuration:

Select this radio button when you want to specify a particular SSL configuration for use when SSL is to be used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

Data type

Radio button

SSL configuration:

The SSL configuration to use when SSL is to be used to secure network communications with the WebSphere queue manager or queue-sharing group.

This property is disabled if the **Centrally managed** radio button is selected and the WebSphere MQ messaging provider resource has been explicitly defined.

This property is always enabled if the WebSphere MQ messaging provider resource is based on a CCDT.

If this WebSphere MQ messaging provider resource is based on a CCDT, this parameter is only used if the relevant entries in the CCDT have been configured to use SSL.

Additionally, if an SSL configuration of `none` is selected, the default centrally managed WebSphere Application Server SSL configuration for the WebSphere MQ messaging provider is used.

Data type Drop-down list

Advanced:

Client ID:

The client identifier to specify when connecting to the WebSphere MQ messaging provider.

Data type String

Provider version:

The WebSphere MQ messaging provider version. This is used to determine whether to connect to a particular version of a queue manager. It is also used to determine the type of functions required by the client.

Data type String

Range The value entered must either be the empty string or be in one of the following formats:

n.n.n.n

n.n.n

n.n

n

where n is a numeric value greater than or equal to zero.

For example 6.0.0.0.

Support distributed two-phase commit protocol:

An option that specifies whether the queue connection factory supports XA coordination of messaging transactions. Enable this option if multiple resources, including this queue connection factory, are to be used in the same transaction.

If you clear this property, you disable support for distributed two-phase commit protocol. The JMS session can still be enlisted in a transaction, but it uses the resource manager local transaction calls session.commit and session.rollback, instead of XA calls. This can lead to an improvement in performance. However, only a single resource can be enlisted in a transaction in WebSphere Application Server.

Last participant support enables you to enlist one non-XA resource with other XA-capable resources.

Data type Check box

Default Selected

Range

Selected
The queue connection factory supports the use of distributed two-phase commit protocols for the coordination of transacted work.

Cleared
The queue connection factory does not support the use of distributed two-phase commit protocols for the coordination of transacted work.

Keep this option selected if transactions involve other resources, including other queues or topics. Clear this option only when you are certain that the queue manager that is the target for this queue connection factory is the only resource in the transaction.

Security settings:

Authentication alias for XA recovery:

The user name and password to use when connecting to WebSphere MQ during XA recovery.

Data type Drop-down list
Default (none)
Range All authentication aliases defined to the cell and the value "(none)", which specifies that no credentials are passed to WebSphere MQ during XA recovery.

Mapping-configuration alias:

This field is used only in the absence of a login configuration on the component resource reference.

When the resource authority value is "container", the preferred way to define the authentication strategy is by specifying a login configuration and associated properties on the component resource reference.

If the **DefaultPrincipalMapping** login configuration is specified, the associated property will be a JAAS - J2C authentication data entry alias. To configure authentication information for use with the queue connection factory, under Related Items, click **JAAS - J2C authentication data** .

Data type Drop-down list
Default (none)
Range The following options are available:
- ClientContainer
- WSLogin
- WSKRB5Login
- DefaultPrincipalMapping
- TrustedConnectionMapping
- KerberosMapping

Container-managed authentication alias:

The authentication alias which specifies the user name and password to use when connecting to the WebSphere MQ messaging provider.

Data type Drop-down list
Default (none)

Range All authentication aliases defined to the cell and the value "(none)", which specifies that no credentials are passed to WebSphere MQ.

WebSphere MQ messaging provider queue connection factory advanced properties:

Use this panel to view or change the advanced properties of the selected queue connection factory for use with the WebSphere MQ messaging provider. These advanced properties control the behavior of connections made to WebSphere MQ messaging provider destinations.

To view WebSphere MQ queue connection factory advanced properties, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Queue connection factories** to display existing queue connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the queue connection factories are defined. Setting the scope in this way restricts the set of queue connection factories displayed.
3. Select the name of the queue connection factory that you want to work with.
4. In the content pane, under Additional properties, click **Advanced properties** to view a list of the advanced properties of the WebSphere MQ queue connection factory.

Under General Properties there are six groups of advanced properties:
- Client reconnect
- Message compression
- Temporary destinations
- Connection consumer
- Message format
- Additional

Make any required changes to these groups and then click **Apply** to return to the queue connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, except for channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring JMS resources for WebSphere MQ. For more information about configuring JMS resources for WebSphere MQ , see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ queue connection factory has the following advanced properties:

Client reconnect options:

This property specifies whether a client mode connection reconnects automatically, or not, in the event of a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment.

Data type Drop-down list
Default DISABLED

Range

DISABLED
Client reconnection does not automatically occur.

ASDEF The value from the DefRecon attribute from the channels stanza of the client configuration file is used. If there is no DefRecon value specified in the client configuration file, ASDEF has the same effect as DISABLED.

QMGR Reconnection occurs only to the queue manager to which the connection was originally connected.

RECONNECT
Reconnection occurs to any queue manager that is consistent with the value of the queue manager attribute. This queue manager might be a different queue manager from the one to which the connection was originally connected.

For more information about automatic client reconnection, see the WebSphere MQ information center.

Client reconnect timeout:

The maximum number of seconds that a client mode connection spends attempting to automatically reconnect to a queue manager after a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment. Whether this parameter is used or not depends on the value of the client reconnect options parameter.

Data type Integer
Units Seconds
Default 1800 (30 minutes)
Range A value greater than zero and up to 2147483647

For more information about automatic client reconnection, see the WebSphere MQ information center.

Compress message headers:

An option that enables the compression of message headers.

Data type Check box
Default Cleared
Range

Cleared
Do not compress message headers.

Selected
Compress message headers.

Compression algorithm for message payloads:

The compression algorithm that is used to compress message payloads.

Data type Drop-down list
Default NONE

Range

RLE Message data compression is performed using run-length encoding.

ZLIBFAST Message data compression is performed using ZLIB encoding with speed prioritized.

ZLIBHIGH Message data compression is performed using ZLIB encoding with compression prioritized.

NONE No message data compression is performed.

WebSphere MQ model queue name:

The model queue that is used as a basis for temporary queue definitions.

Data type String
Default SYSTEM.DEFAULT.MODEL.QUEUE

Temporary queue prefix:

The prefix that is attached to the beginning of the names generated for temporary queues.

Data type String

Retain messages, even if no matching consumer is available:

An option that determines whether messages for which there is no matching consumer are retained on the input queue or dealt with according with their disposition options.

Data type Check box
Default Selected
Range

Cleared Do not retain messages.

Selected Retain messages.

Polling interval:

This setting is applicable in the client container only. When using a WebSphere MQ Version 6 queue manager (or WebSphere MQ Version 5.3 for z/OS), this setting configures the mechanism used to dispatch messages to JMS asynchronous consumers.

This setting is used when the set of WebSphere MQ queues that is being asynchronously consumed from exceeds the number of threads that are available internally to synchronously get messages from the WebSphere MQ queue. The setting determines how long a thread waits for a message to arrive at a WebSphere MQ queue before polling another WebSphere MQ queue in the set.

Data type Integer
Units Milliseconds
Default 5000
Range A value greater than zero.

Rescan interval:

When using a WebSphere MQ Version 6 queue manager (or WebSphere MQ Version 5.3 for z/OS), this setting configures the mechanism used to dispatch messages to JMS asynchronous consumers.

This setting is used when the set of WebSphere MQ queues that is being asynchronously consumed from exceeds the number of threads that are available internally to synchronously get messages from the WebSphere MQ queue. The setting determines how long a thread retrieves messages from a WebSphere MQ queue before switching to use messages from another WebSphere MQ queue in the set.

Data type	Integer
Units	Milliseconds
Default	5000
Range	A value greater than zero.

Maximum batch size:

The maximum number of messages that can be removed from a queue before at least one must be delivered to an asynchronous consumer.

Data type	Integer
Default	10
Range	A value greater than zero.

Coded character set identifier:

The character set to use when you are encoding strings in the message.

Data type	Integer
Default	819
Range	A value greater than zero. The coded character set identifier (CCSID) must be one of the CCSIDs supported by WebSphere MQ.

For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *System Administration* and the *Application Programming Reference* sections of the WebSphere MQ information center.

Append an RFH version 2 header to reply messages:

The action required when sending a reply message to the reply-to queue obtained from a message that does not include an RFH version 2 header.

Data type	Check box
Default	Cleared
Range	**Selected** Append an RFH version 2 header to reply messages regardless of whether the original message had an RFH version 2 header. **Cleared** Append an RFH version 2 header to reply messages only if the original message had an RFH version 2 header.

Fail JMS method calls if the WebSphere MQ queue manager is quiescing:

An option that enables selected JMS operations to fail when the queue manager is put into a quiescing state. Selecting this option enables the queue manager to quiesce successfully and shut down.

Data type Check box
Default Selected
Range
Cleared
Do not fail JMS operations if the queue manager is quiescing.
Selected
Fail JMS operations if the queue manager is quiescing.

WebSphere MQ messaging provider connection factory client transport settings:

Use this panel to view or change the client transport settings of a connection factory, queue connection factory, or topic connection factory for use with the WebSphere MQ messaging provider. Client transport properties affect how a client connection is established with a WebSphere MQ queue manager or queue-sharing group. Updates to the settings take effect when the server is restarted.

To view WebSphere MQ connection factory, queue connection factory, or topic connection factory client transport settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS**.
2. Click **Connection factories**, **Queue connection factories**, or **Topic connection factories** to display existing connection factories, queue connection factories or topic connection factories.
3. Select the name of the connection factory, queue connection factory, or topic connection factory that you want to work with.
4. In the content pane under Additional Properties, click **Client transport properties** to view a list of the client transport settings of the WebSphere MQ connection factory, queue connection factory, or topic connection factory.

Under General Properties there are two groups of properties:
- Additional SSL settings (for more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *Using Java* section of the WebSphere MQ information center)
- Channel exits

Make any required changes to these groups and then click **Apply** to return to the connection factory, queue connection factory or topic connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ connection factory, queue connection factory, or topic connection factory has the following client transport settings properties:

Certificate revocation list:

A list of LDAP URLs pointing to LDAP repositories of SSL certificates that might have been revoked.

Data type String
Default No certificate revocation list
Range The value must be a space-separated list of LDAP URLs.

Peer name:

A name (possibly including wildcards) that must match the distinguished name of the peer SSL certificate for a connection to be established.

Data type	String
Default	Do not check the distinguished name of the peer certificate.
Range	Validated according to the rules for a WebSphere MQ SSLPEER channel parameter.

Reset count:

The total number of bytes to transfer over an SSL connection before renegotiating the symmetric encryption keys used to secure the connection.

Data type	Integer
Default	0 (do not renegotiate)
Range	The value must be in the range 0 through 999,999,999 (inclusive).

Receive exits:

A comma-separated list of Java class names corresponding to receive exits to be loaded.

Data type	String

Receive exit initialization data:

Initialization data to be passed to the receive exit.

Data type	String

Send exits:

A comma-separated list of Java class names corresponding to send exits to be loaded.

Data type	String

Send exit initialization data:

Initialization data to be passed to the send exit.

Data type	String

Security exit:

A Java class name corresponding to the security exit to be loaded.

Data type	String

Security exit initialization data:

Initialization data to be passed to the security exit.

Data type	String

Connection pool settings:

Use this page to configure connection pool settings.

This administrative console page is common to JDBC data sources and JMS connection factories (unified, queue, or topic connection factories). To view this page, the path depends on the type of resource, but generally you select an instance of the resource type then click **Connection Pool**. For example:

- Click **Resources** > **JDBC** > **Data Sources** > ***data_source*** > **[Additional Properties] Connection pool properties**
- Click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Connection pool**

Note: Connection pooling is not supported in an application client. The application client calls the database directly and does not go through a data source. If you want to use the getConnection() request from the application client, configure the JDBC provider in the application client deployment descriptors, using Rational Application Developer or an assembly tool. The connection is established between application client and the database. Application clients do not have a connection pool, but you can configure JDBC provider settings in the client deployment descriptors.

Connection timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

This value indicates the number of seconds that a connection request waits when there are no connections available in the free pool and no new connections can be created. This usually occurs because the maximum value of connections in the particular connection pool has been reached.

For example, if Connection timeout is set to 300, and the maximum number of connections are all in use, the pool manager waits for 300 seconds for a physical connection to become available. If a physical connection is not available within this time, the pool manager initiates a `ConnectionWaitTimeout` exception. In most cases, you should not retry the getConnection() method; if a longer wait time is required you should increase the Connection timeout setting value. If a `ConnectionWaitTimeout` exception is caught by the application, review the expected connection pool usage of the application and tune the connection pool and database accordingly.

If the Connection timeout is set to 0, the pool manager waits as long as necessary until a connection becomes available. This happens when the application completes a transaction and returns a connection to the pool, or when the number of connections falls below the value of Maximum Connections, and a new physical connection is created.

If Maximum Connections is set to 0, an infinite number of physical connections are enabled, and the Connection timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Maximum connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. When this number is reached, no new physical connections are created. The requester waits until a physical connection that is currently in use returns to the pool, or until a ConnectionWaitTimeoutException error displays. For example, if the Max Connections value is set to 5, and there are 5 physical connections in use, the pool manager waits for the amount of time specified in Connection timeout for a physical connection to become free.

Knowing the number of connection pools that can potentially request connections from the backend, such as a DB2 database or a CICS server, helps you determine a value for the Maximum Connections property.

For multiple stand-alone application servers that use the same data source configuration, or J2C connection factory configuration, a separate physical connection pool exists for each server. If you clone these same application servers, WebSphere Application Server (base) implements a separate connection pool for each clone.

All of these connection pools correspond to the same data source or connection factory configuration. Therefore all of these connection pools can potentially request connections from the same backend resource, at the same time. The single Maximum Connections value that you set on this console panel applies to every one of these connection pools. Consequently, setting a high Maximum Connections value can result in a load of connection requests that overwhelms your backend resource.

Data type	Integer
Default	10
Range	0 to maximum integer If Max Connections is set to 0, the Connection timeout value is ignored.

Tip: For better performance, set the value for the connection pool lower than the value for the maximum thread pool connections of the web container. To configure this setting click **Servers** > **Server types** > **WebSphere application servers** > ***server*** > **Thread Pools**, and modify the web container property. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the processor load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Minimum connections:

Specifies the minimum number of physical connections to maintain.

If the size of the connection pool is at or below the minimum connection pool size, the Unused timeout thread does not discard physical connections. However, the pool does not create connections solely to ensure that the minimum connection pool size is maintained. Also, if you set a value for Aged timeout, connections with an expired age are discarded, regardless of the minimum pool size setting.

For example, if the Minimum Connections value is set to 3, and one physical connection is created, the Unused timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the Minimum Connections setting.

Data type	Integer
Default	1
Range	0 to max int

Reap time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if Reap Time is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the Unused timeout and Aged timeout settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused timeout and Aged timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused timeout, until it reaches the number of connections specified in Minimum Connections. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread, set Reap Time to `0`, or set both Unused timeout and Aged timeout to `0`. The recommended way to disable the pool maintenance thread is to set Reap Time to `0`, and Unused timeout and Aged timeout are ignored. However, if Unused timeout and Aged timeout are set to `0`, the pool maintenance thread runs, but only physical connections that timeout, because of non-zero timeout values, are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused timeout value higher than the Reap timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections exceeds the Minimum Connections setting. For example, if the unused timeout value is set to `120`, and the pool maintenance thread is enabled (Reap Time is not `0`), any physical connection that remains unused for 2 minutes is discarded.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting Aged timeout to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged timeout value higher than the Reap timeout value for optimal performance.

For example, if the Aged timeout value is set to `1200`, and the Reap Time value is not `0`, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. The only

exception is if the connection is involved in a transaction when the aged timeout is reached, the application server will not discard the connection until after the transaction is completed and the connection is closed.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**.

Data type	String
Defaults	• EntirePool for J2C connection factories and JMS-related connection factories • EntirePool for WebSphere Version 4.0 data sources • EntirePool for current version data sources that you create through the administrative console • EntirePool for current version data sources that you script through wsadmin AdminConfig commands, starting JDBC templates that are built into WebSphere Application Server. For information about the command createUsingTemplate, see the topic, Commands for the AdminConfig object. • FailingConnectionOnly for data sources that you script in wsadmin tool without starting JDBC templates :

Range

EntirePool
All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and issues a stale connection Exception during the next operation on that connection. Subsequent getConnection() requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this closure is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice.

FailingConnectionOnly
Only the connection that caused the stale connection exception is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a possibility that the next getConnection() request from the application can return a connection from the pool that is also stale. The result is more stale connection exceptions.

The connection pretest function attempts to insulate an application from pooled connections that are not valid. When a backend resource, such as a database, goes down, pooled connections that are not valid might exist in the free pool. This is especially true when the purge policy is failingConnectionOnly; in this case, the failing connection is removed from the pool. Depending on the failure, the remaining connections in the pool might not be valid.

Session pool settings:

Use this page to configure session pool settings.

This administrative console page is common to JMS unified connection factories, queue connection factories and topic connection factories. To view this page, you select an instance of the resource type then click **Session pools**. For example, click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Session pools**.

Connection Timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

The wait is necessary when the maximum value of connections (**Max Connections**) to a particular connection pool is reached . For example, if *Connection Timeout* is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is *not* available within this time, the Pool Manager throws a ConnectionWaitTimeoutException. It usually does not make sense to retry the getConnection() method, because if a longer wait time is required, you should set the **Connection Timeout** setting to a higher

value. Therefore, if this exception is caught by the application, the administrator should review the expected usage of the application and tune the connection pool and the database accordingly.

If Connection Timeout is set to 0, the Pool Manager waits as long as necessary until a connection is allocated (which happens when the number of connections falls below the value of **Max Connections**).

If Max Connections is set to 0, which enables an infinite number of physical connections, then the Connection Timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Max Connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use returns to the pool, or a ConnectionWaitTimeoutException is thrown.

For example, if the Max Connections value is set to 5, and there are five physical connections in use, the pool manager waits for the amount of time specified in Connection Timeout for a physical connection to become free.

If Max Connections is set to 0, the Connection Timeout value is ignored.

For better performance, set the value for the connection pool lower than the value for the Max Connections option in the web container. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

If clones are used, one data pool exists for each clone. Knowing the number of data pools is important when configuring the database maximum connections.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the CPU load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Data type	Integer
Default	10
Range	0 to max int

Min Connections:

Specifies the minimum number of physical connections to maintain.

Until this number is reached, the pool maintenance thread does not discard physical connections. However, no attempt is made to bring the number of connections up to this number. If you set a value for Aged Timeout, the minimum is not maintained. All connections with an expired age are discarded.

For example if the **Min Connections** value is set to 3, and one physical connection is created, the Unused Timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the **Min Connections** setting.

Data type	Integer
Default	1
Range	0 to max int

Reap Time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if **Reap Time** is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in **Min Connections**. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread set Reap Time to 0, or set both Unused Timeout and Aged Timeout to 0. The recommended way to disable the pool maintenance thread is to set Reap Time to 0, in which case Unused Timeout and Aged Timeout are ignored. However, if Unused Timeout and Aged Timeout are set to 0, the pool maintenance thread runs, but only physical connections that timeout due to non-zero timeout values are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused Timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the **Min Connections** setting. For example, if the unused timeout value is set to 120, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the **Reap Time** value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged Timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting **Aged Timeout** to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged Timeout value higher than the **Reap Timeout** value for optimal performance. For example, if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that

remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, are affected by the Reap Time value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge Policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**. Java EE Connector Architecture (JCA) data sources can have either option. WebSphere Version 4.0 data sources always have a purge policy of **EntirePool**.

Data type	String
Default	FailingConnectionOnly
Range	**EntirePool** All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and throws a *StaleConnectionException* during the next operation on that connection. Subsequent *getConnection* requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice. **FailingConnectionOnly** Only the connection that caused the *StaleConnectionException* is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a good possibility that the next *getConnection* request from the application can return a connection from the pool that is also stale, resulting in more stale connection exceptions.

Configuring a topic connection factory for the WebSphere MQ messaging provider

Use this task to view or change the configuration of a JMS topic connection factory for publish/subscribe messaging with the WebSphere MQ messaging provider.

About this task

This task applies to topic connection factories. With JMS 1.1, domain-independent (unified) connection factories are preferred to domain-specific queue connection factories and topic connection factories. To view or configure a unified connection factory, see “Configuring a unified connection factory for the WebSphere MQ messaging provider” on page 761.

To view or change the configuration of an existing topic connection factory for use with the WebSphere MQ messaging provider, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Resources** > **JMS->Topic connection factories**. A list of existing topic connection factories, with a summary of their properties, is displayed.
2. Select the **Scope** corresponding to the scope at which the topic connection factory is visible to applications.
3. Click the name of the topic connection factory that you want to view or change. Configuration details for the topic connection factory are displayed.
4. Under **General properties** make any required changes.

   For information about each of the available fields, see “WebSphere MQ messaging provider topic connection factory settings.”
5. Click **Apply** to save the configuration.
6. Optional: Click **Advanced properties** to display or change the list of advanced properties of your topic connection factory.
7. Optional: Click **Broker properties** to display or change the list of broker properties of your topic connection factory.
8. Optional: Click **Custom properties** to display or change the list of custom properties of your topic connection factory.
9. Optional: Click **Client transport properties** to display or change the list of client transport properties of your topic connection factory. This link is only appears if, when you created the connection factory, you chose to enter all the information required to connect to WebSphere MQ. This link does not appear if you have chosen to use a client channel definition table (CCDT) to establish a connection to WebSphere MQ
10. Optional: Click **Connection pools** to display or change the connection pools detail of your topic connection factory.
11. Optional: Click **Session pools** to display or change the session pools detail of your topic connection factory.

    There is a mechanism (session.sharing.scope custom property) to make JMS sessions unshareable. This means that whenever an application calls Session.close(), the JMS session is automatically released from any transaction that is associated with and returned to the session pool. This means that sessions can be cleaned up and removed from the session pool even if the servlet or asynchronous bean that created it is still running.
12. Click **OK**.
13. Save your changes to the master configuration.
14. To have the changed configuration take effect, stop then restart the application server.

WebSphere MQ messaging provider topic connection factory settings

Use this panel to view or change the configuration properties of the selected topic connection factory for use with the WebSphere MQ messaging provider. These configuration properties control how connections are created to the associated JMS topic destination.

A WebSphere MQ topic connection factory is used to create JMS connections to topic destinations provided by WebSphere MQ for publish/subscribe messaging.

To view WebSphere MQ topic connection factory settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Topic connection factories** to display existing topic connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the topic connection factories are defined. This restricts the set of topic connection factories displayed.
3. Select the name of the topic connection factory that you want to work with.

Under General Properties there are four groups of properties:
- "Administration"
- "Connection" on page 816
- "Advanced" on page 822
- "Security settings" on page 823

Make any required changes to the Administration, Connection, Advanced and Security settings groups of properties, and then click **Apply** to save the configuration before, in the content pane under Additional Properties, you click any of the following links:
- **Advanced properties** to display the advanced properties of your WebSphere MQ topic connection factory.
- **Broker properties** to display the broker properties of your WebSphere MQ topic connection factory.
- **Custom properties** to display the custom properties of your WebSphere MQ topic connection factory.
- **Client transport properties** to display or change the client transport properties of your WebSphere MQ topic connection factory. If the selected topic connection factory was not created using a Client Channel Definition Table (CCDT), follow this link to enter all the information required to connect to WebSphere MQ. If the selected topic connection factory was created using a CCDT, you do not need to supply the client transport properties, and so the link is absent.
- **Connection pools** to display the connection pools detail of your WebSphere MQ topic connection factory
- **Session pools** to display the session pools detail of your WebSphere MQ topic connection factory.

Under Related Items, you can click **JAAS - J2C authentication data** to configure authentication information for use with the topic connection factory.

You can specify the following additional properties by using WebSphere MQ administrative commands:
- **-localAddress**
- **-clonedSubs**
- **-componentAuthAlias**

For more information about these properties, refer to the "createWMQConnectionFactory command" on page 888.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when you configured the JMS resources in WebSphere MQ. For more information about configuring JMS resources in WebSphere MQ, see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ topic connection factory has the following properties.

Administration:

Scope:

The level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource at a different scope, change the scope on the WebSphere MQ topic connection factory collection panel, then click **Apply**, before clicking the link for the type of resource.

Data type	String

Provider:

The JMS provider assigned when the topic connection factory is created.

For all topic connection factories created using this panel, the provider is the WebSphere MQ messaging provider.

The provider is displayed for information only.

Data type	String

Name:

The name by which this topic connection factory is known for administrative purposes within WebSphere Application Server.

Data type	String
Range	The name must be unique within the set of JMS topic connection factories defined to the cell.

JNDI name:

The JNDI name that is used to bind the topic connection factory into the JNDI namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

Data type	String

Description:

A description of this topic connection factory for administrative purposes within WebSphere Application Server.

Data type	String
Default	Null

Connection:

The information required to configure a connection depends on whether the selected topic connection factory was created using a Client Channel Definition Table (CCDT).

If the selected topic connection factory was created using a CCDT, only the following properties are displayed:

- Client channel definition table URL
- Queue manager
- SSL configuration

If the selected topic connection factory was not created using a CCDT, the following properties are displayed:

- Queue manager
- Transport
- If **Enter host and port information in the form of separate host and port values** is selected, the connection name list property cannot be used and the following properties can be used:
  - Host name
  - Port
- If **Enter host and port information in the form of a connection name list** is selected, the connection name list property can be used and the following properties cannot be used:
  - Host name
  - Port
- Server connection channel
- If you clear the check box for the **Use SSL to secure communication with Websphere MQ** property, the following properties cannot be used:
  - Centrally managed
  - Specific configuration
  - SSL configuration

For more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *Using Java* section of the WebSphere MQ information center.

Client channel definition table URL:

A URL that specifies the location of a WebSphere MQ CCDT.

Data type String

Queue manager:

If the specified topic connection factory is based on a CCDT, this property is used to select an entry in the CCDT. Otherwise, it is the name of the queue manager or queue-sharing group to connect to. A connection is established to this WebSphere MQ resource to send or receive messages.

Data type String

Range

If this topic connection factory is not based on a CCDT, the value must be a valid queue manager name.

If this topic connection factory is based on a CCDT, the value must be one of the following:

- A valid queue manager name
- An asterisk (*) followed by the name of a queue manager group[1]
- An asterisk (*)
- Blank[1]

[1]When you specify the value of the Queue manager property in this form in combination with a CCDT, individual connections established by using the topic connection factory might connect to different queue managers. Selection from multiple queue managers occurs when the CCDT contains multiple client connection channel definitions with a matching queue manager name (QMNAME) parameter, and these connection channel definitions define the network connection details of different queue managers.

If the specified connection factory is based on a CCDT, and the CCDT can select from more than one queue manager, you might not be able to recover global transactions. Therefore, for connection factories that specify a CCDT, you have two alternatives:

- Avoid any ambiguity about the target queue manager when specifying the Queue manager property, which means that the specified value of this property must not include an asterisk (*).
- Avoid using the resources with applications that enlist in global transactions.

Transport:

The WebSphere MQ transport type for the connection. The transport type is used to determine the exact mechanisms used to connect to WebSphere MQ.

Data type Drop-down list
Default **Bindings, then client**

Range

Client Use a TCP/IP-based network connection to communicate with the WebSphere MQ queue manager

Bindings, then client
Attempt a bindings mode connection to the queue manager. If this is not possible, revert to the client transport.

Bindings
Establish a cross-memory connection to a queue manager running on the same node. The following Client Transport Mode properties are disabled:
- Hostname
- Port
- Connection name list
- Server connection channel

For more information about configuring a transport type of *Bindings, then client* or *Bindings*, refer to "Configuring the WebSphere MQ messaging provider with native libraries information" on page 723.

Enter host and port information in the form of separate host and port values:

If this radio button is selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the host name and port properties.

Selecting this option enables the host name and port properties, and disables the connection name list property. To enter connection name list information, click **Enter host and port information in the form of a connection name list**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type Radio button

Hostname:

The host name, IPv4, or IPv6 address of the WebSphere MQ queue manager to connect to.

Data type String

Port:

The port number on which WebSphere MQ is listening.

Data type Integer
Default 1414
Range The value must be in the range 1 to 65536 (inclusive).

Enter host and port information in the form of a connection name list:

If this radio button is selected, this means that the connection to the WebSphere MQ queue manager is made using the information supplied by the connection name list property.

Connection name lists can be used to connect to a single queue manager or to a multi-instance queue manager. For more information on using a multi-instance queue manager, see the WebSphere MQ information centre. Selecting this option enables the connection name list property and disables the host name and port properties. To enter separate host and port information, click **Enter host and port information in the form of separate host and port values**.

This radio button is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

Data type	Radio button

Connection name list:

The connection name list specifying the host name and port details to use when you want the connection factory to connect to a multi-instance queue manager.

This property must only be used to allow connection to a multi-instance queue manager. It must not be used to allow connections to non-multi-instance queue managers as that can result in transaction integrity issues.

Data type	String
Default	Unset
Range	This field must be set to a string in the following form: *host*[(*port*)][,*host*(*port*)] The port information is optional, and if not specified, defaults to 1414. *host* must be a valid TCP/IP host name or IPv4 or IPv6 address. *port* must be an integer value in the range 1 to 65536 (inclusive). For example: `localhost(1234),remotehost1(1234),remotehost2`

When the connection name list property is specified, the host name or port properties are automatically set to the host name and port number of the first entry in the connection name list. So if you specified `localhost(1234),remotehost1(1234),remotehost2`, the host name would be set to localhost and port would be set to 1234.

This property is only available if the scope property specifies a cell, or if the scope property specifies a node or server and that node or server is running WebSphere Application Server Version 8.0.

When used in a mixed cell environment, the information in the connection name list property is not available to versions of WebSphere Application Server earlier than Version 8.0. In this case, the information in the host name and port name fields, based on the first element in the connection name list, is used instead.

Server connection channel:

The WebSphere MQ server connection channel name used when connecting to WebSphere MQ.

Data type	String
Default	SYSTEM.DEF.SVRCONN

Range

The value must be a server connection channel defined to the WebSphere MQ queue manager that is being connected to.

Use SSL to secure communications with WebSphere MQ:

This option determines whether the SSL (Secure Sockets Layer) protocol is used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

When using a WebSphere MQ messaging provider topic connection factory in the application server environment, the application server manages SSL configuration. To change SSL configuration parameters, use the administrative console to navigate to the **Security** > **SSL certificate and key management** panel.

When using a WebSphere MQ messaging provider topic connection factory in the client environment, the client takes SSL configuration information from the ssl.client.props file. Use of this file is detailed in the related reference information for this topic.

You can only use one cipher suite in the SSL configuration for a WebSphere MQ messaging provider topic connection factory. If you specify more than one cipher suite, only the first one is used.

Data type

Check box. If this check box is cleared, the following SSL properties are disabled:

- Centrally managed
- Specific configuration
- SSL configuration

Centrally managed:

When the SSL protocol is used to communicate with WebSphere MQ, select this radio button to specify that the SSL configuration is taken from the centrally managed WebSphere Application Server SSL configuration.

When you select this radio button, the host name and port attributes from the WebSphere MQ messaging provider connection factory are used to select an appropriate SSL configuration. If host and port information has been supplied by a connection name list this means that the host name and port information of the first element in the list are used to select an appropriate SSL configuration. To provide the SSL configuration which will be matched to the connection factory, see the Dynamic outbound endpoint SSL configuration settings topic listed under related reference.

Data type

Radio button

Specific configuration:

Select this radio button when you want to specify a particular SSL configuration for use when SSL is to be used to secure network communications with the WebSphere MQ queue manager or queue-sharing group.

Data type

Radio button

SSL configuration:

The SSL configuration to use when SSL is to be used to secure network communications with the WebSphere MQ queue manager or queue sharing group.

This property is disabled if the **Centrally managed** radio button is selected and the WebSphere MQ messaging provider resource has been explicitly defined.

This property is always enabled if the WebSphere MQ messaging provider resource is based on a CCDT.

If this WebSphere MQ messaging provider resource is based on a CCDT, this parameter is only used if the relevant entries in the CCDT have been configured to use SSL.

Additionally, if an SSL configuration of `none` is selected, the default centrally managed WebSphere Application Server SSL configuration for the WebSphere MQ messaging provider is used.

Data type Drop-down list

Advanced:

Client ID:

The client identifier to specify when connecting to WebSphere MQ messaging provider.

Data type String

Allow cloned durable subscriptions:

An option that determines whether multiple instances of a durable subscription can be accessed concurrently by different servers.

Data type Check box
Default Cleared
Range
Selected
Multiple instances of a durable subscription can be accessed concurrently by different servers.
Cleared
Multiple instances of a durable subscription cannot be accessed concurrently by different servers.

Provider version:

The WebSphere MQ messaging provider version. This is used to determine whether to connect to a particular version of a queue manager. It is also used to determine the type of functions required by the client.

Data type String
Range
The value entered must either be the empty string or be in one of the following formats:

n.n.n.n

n.n.n

n.n

n

where n is a numeric value greater than or equal to zero.

For example 6.0.0.0.

Support distributed two-phase commit protocol:

An option that specifies whether the topic connection factory supports XA coordination of messaging transactions. Enable this option if multiple resources, including this topic connection factory, are to be used in the same transaction.

If you clear this property, you disable support for distributed two-phase commit protocol. The JMS session can still be enlisted in a transaction, but it uses the resource manager local transaction calls session.commit and session.rollback, instead of XA calls. This can lead to an improvement in performance. However, only a single resource can be enlisted in a transaction in WebSphere Application Server.

Last participant support enables you to enlist one non-XA resource with other XA-capable resources.

Data type Check box
Default Selected
Range
Selected
The connection factory supports the use of distributed two-phase commit protocols for the coordination of transacted work.
Cleared
The connection factory does not support the use of distributed two-phase commit protocols for the coordination of transacted work.

Do not enable XA when the queue manager specified by the topic connection factory is the only resource in the transaction. Enable XA if transactions involve other resources, including other queues or topics.

Security settings:

Authentication alias for XA recovery:

The user name and password to use when connecting to WebSphere MQ during XA recovery.

Data type Drop-down list
Default (none)
Range All authentication aliases defined to the cell and the value "(none)", which specifies that no credentials are passed to WebSphere MQ during XA recovery.

Mapping-configuration alias:

This field is used only in the absence of a login configuration on the component resource reference.

When the resource authority value is "container", the preferred way to define the authentication strategy is by specifying a login configuration and associated properties on the component resource reference.

If the **DefaultPrincipalMapping** login configuration is specified, the associated property will be a JAAS - J2C authentication data entry alias. To configure authentication information for use with the topic connection factory, under Related Items, click **JAAS - J2C authentication data** .

Data type Drop-down list
Default (none)

Range

The following options are available:

ClientContainer
WSLogin
WSKRB5Login
DefaultPrincipalMapping
TrustedConnectionMapping
KerberosMapping

Container-managed authentication alias:

The authentication alias which specifies the user name and password to use when connecting to the WebSphere MQ messaging provider.

Data type Drop-down list
Default (none)
Range All authentication aliases defined to the cell and the value "(none)", which specifies that no credentials are passed to WebSphere MQ.

WebSphere MQ messaging provider topic connection factory advanced properties:

Use this panel to view or change the advanced properties of the selected topic connection factory for use with the WebSphere MQ messaging provider. These advanced properties control the behavior of connections made to WebSphere MQ messaging provider destinations.

To view WebSphere MQ topic connection factory advanced properties, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->Topic connection factories** to display existing topic connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the topic connection factories are defined. Setting the scope in this way restricts the set of queue connection factories displayed.
3. Select the name of the topic connection factory that you want to work with.
4. In the content pane, under Additional properties, click **Advanced properties** to view a list of the advanced properties of the WebSphere MQ topic connection factory.

Under General Properties there are six groups of advanced properties:

- Client reconnect
- Message compression
- Temporary destinations
- Connection consumer
- Message format
- Additional

Make any required changes to these groups and then click **Apply** to return to the topic connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:

- Names can have a maximum of 48 characters, except for channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring JMS resources for WebSphere MQ. For more information about configuring JMS resources for WebSphere MQ, see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ topic connection factory has the following advanced properties.

Client reconnect options:

This property specifies whether a client mode connection reconnects automatically, or not, in the event of a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment.

Data type Drop-down list
Default DISABLED
Range
DISABLED
Client reconnection does not automatically occur.
ASDEF The value from the DefRecon attribute from the channels stanza of the client configuration file is used. If there is no DefRecon value specified in the client configuration file, ASDEF has the same effect as DISABLED.
QMGR Reconnection occurs only to the queue manager to which the connection was originally connected.
RECONNECT
Reconnection occurs to any queue manager that is consistent with the value of the queue manager attribute. This queue manager might be a different queue manager from the one to which the connection was originally connected.

For more information about automatic client reconnection, see the WebSphere MQ information center.

Client reconnect timeout:

The maximum number of seconds that a client mode connection spends attempting to automatically reconnect to a queue manager after a communications or queue manager failure. This parameter is ignored unless the connection factory is being used in a thin or managed client environment. Whether this parameter is used or not depends on the value of the client reconnect options parameter.

Data type Integer
Units Seconds
Default 1800 (30 minutes)
Range A value greater than zero and up to 2147483647

For more information about automatic client reconnection, see the WebSphere MQ information center.

Compress message headers:

An option that enables the compression of message headers.

Data type Check box
Default Cleared
Range
Cleared
Do not compress message headers.
Selected
Compress message headers.

Compression algorithm for message payloads:

The compression algorithm to use to compress message payloads.

Data type Drop-down list
Default NONE
Range
RLE Message data compression is performed using run-length encoding.
ZLIBFAST Message data compression is performed using ZLIB encoding with speed prioritized.
ZLIBHIGH Message data compression is performed using ZLIB encoding with compression prioritized.
NONE No message data compression is performed.

Temporary topic prefix:

The prefix to append to the beginning of the names generated for temporary topics.

Data type String

Polling interval:

This setting is applicable in the client container only. When using a WebSphere MQ Version 6 queue manager (or WebSphere MQ Version 5.3 for z/OS), this setting configures the mechanism used to dispatch messages to JMS asynchronous consumers.

This setting is used when the set of WebSphere MQ queues that is being asynchronously consumed from exceeds the number of threads that are available internally to synchronously get messages from the WebSphere MQ queue. The setting determines how long a thread waits for a message to arrive at a WebSphere MQ queue before polling another WebSphere MQ queue in the set.

Data type Integer
Units Milliseconds
Default 5000
Range A value greater than zero.

Maximum batch size:

The maximum number of messages that can be removed from a queue before at least one must be delivered to an asynchronous consumer.

Data type Integer
Default 10
Range A value greater than zero.

Coded character set identifier:

The character set to use when you are encoding strings in the message.

Data type Integer
Default 819
Range A value greater than zero. The coded character set identifier (CCSID) must be one of the CCSIDs supported by WebSphere MQ.

For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *System Administration* and the *Application Programming Reference* sections of the WebSphere MQ information center.

Fail JMS method calls if the WebSphere MQ messaging provider queue manager is quiescing:

An option that enables selected JMS operations to fail when the queue manager is put into a quiescing state. Selecting this option enables the queue manager to quiesce successfully and shut down.

Data type Check box
Default Selected
Range
Cleared
Do not fail JMS operations if the queue manager is quiescing.
Selected
Fail JMS operations if the queue manager is quiescing.

WebSphere MQ messaging provider connection factory broker properties:

Use this panel to view or change the broker settings of the selected connection factory, or topic connection factory, for use with the WebSphere MQ messaging provider. These broker settings determine how the WebSphere MQ messaging provider interacts with a broker for the purposes of publishing messages and subscribing to topics. Updates to the settings take effect when the server is restarted.

To view WebSphere MQ connection factory, or topic connection factory, broker properties use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Connection factories** to display existing connection factories, or click **Resources** > **JMS->Topic connection factories** to display existing topic connection factories.
2. If appropriate, in the content pane, change the **Scope** setting to the level at which the connection factories are defined. This restricts the set of connection factories displayed.
3. Select the name of the connection factory, or topic connection factory, that you want to work with.
4. In the content pane, under Additional Properties, click **Broker properties** to display the broker properties of the WebSphere MQ connection factory, or topic connection factory.

Under General Properties there are four groups of properties:
- Queues
- Capabilities
- Tuning
- Additional

Make any required changes to these groups and then click **Apply** to return to the connection factory, or topic connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the WebSphere MQ *Using Java* section of the WebSphere MQ library.

A WebSphere MQ connection factory, or topic connection factory, has the following broker properties:

Broker control queue:

The queue to which broker control messages are sent.

Data type String
Default SYSTEM.BROKER.CONTROL.QUEUE

Broker publication queue:

The queue to which publication messages are sent.

Data type String
Default SYSTEM.BROKER.DEFAULT.STREAM

Broker subscriber queue:

The queue to which subscription messages are sent.

Data type String
Default SYSTEM.JMS.ND.SUBSCRIBER.QUEUE

Broker connection consumer subscription queue:

The queue to which subscription messages that are destined for a connection consumer are sent.

Data type String
Default SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE

Version:

The version of the broker that is used. This determines some of the capabilities that the broker is assumed to have. For example, whether to use an RFH version 1 or version 2 header in publications.

Data type Radio button
Default Version 1 broker
Range

Version 1 broker
Message selection cannot be specified.

Version 2 broker
Message selection can be specified. If you select this option, you must also complete **Specify where message selection occurs**.

Specify where message selection occurs:

The process in which message selection is performed. This property is enabled only if **Version 2 broker** was selected.

Data type Drop-down list
Default CLIENT

Range

CLIENT
Message selection is performed in the application server process.

BROKER
Message selection is performed in the broker process.

Subscription store:

The process for tracking subscriptions.

Data type Drop-down list
Default MIGRATE
Range

MIGRATE
Any information that is held on queues is migrated to the broker mechanism for persisting subscription information. If subscription information is already persisted using the broker mechanism then specifying a value of `Migrate` is equivalent to specifying a value of `Broker`.

BROKER
Internal broker mechanisms are used to track subscription information.

QUEUE
A designated WebSphere MQ queue is used to record information about current subscriptions.

Durable subscription state refresh interval:

How often a long running transaction is recreated and used to clean up durable subscriptions, for some versions of the queue manager.

Data type Integer
Default 60000
Range Any positive integer

Subscription cleanup level:

How aggressively messages are cleaned up if the subscriber that is expected to consume the messages terminates unexpectedly.

Data type Drop-down list
Default SAFE

Range

SAFE A conservative algorithm is used to clean up subscriptions.

ASPROP
The cleanup algorithm is determined by a system property.

NONE No cleanup of subscriptions is performed.

STRONG
An aggressive algorithm is used to clean up subscriptions.

Subscription cleanup interval:

How often to check for orphaned subscriptions and clean up messages.

Data type Integer
Default 3600000
Range Any positive integer

Subscription wildcard format:

The wildcard format used for subscribing to more than one topic in a topic hierarchy.

Data type Drop-down list
Default character wildcards
Range

character wildcards
You can use an asterisk (*) or question mark (?) to represent characters, or strings of characters, in a topic name.

* is interpreted as matching many characters.

? is interpreted as matching a single character.

topic level wildcards
You can use a plus sign (+) or number sign (#) to represent topics in a multilevel topic hierarchy.

+ is interpreted as matching a single topic name.

is interpreted as matching many topics in the hierarchy. / is used to delimit topics.

Publish acknowledgement window:

The number of messages to publish before publishing a message that requires broker acknowledgement

Data type Integer
Default 25
Range Any positive integer

Optimize for sparse subscription patterns:

An option to specify whether this connection factory is anticipated to receive a high proportion of messages that match its selection criteria. This information can be used to optimize message delivery.

Data type Check box
Default Cleared
Range
Cleared
Subscriptions frequently receive matching messages.
Selected
Subscriptions do not frequently receive matching messages.

Broker queue manager:

The name of the queue manager that is running the broker, if it is not the same as the queue manager to which the connection factory connects.

Data type String
Default The queue manager name that was specified in the connection factory.

WebSphere MQ messaging provider connection factory client transport settings:

Use this panel to view or change the client transport settings of a connection factory, queue connection factory, or topic connection factory for use with the WebSphere MQ messaging provider. Client transport properties affect how a client connection is established with a WebSphere MQ queue manager or queue-sharing group. Updates to the settings take effect when the server is restarted.

To view WebSphere MQ connection factory, queue connection factory, or topic connection factory client transport settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS**.
2. Click **Connection factories**, **Queue connection factories**, or **Topic connection factories** to display existing connection factories, queue connection factories or topic connection factories.
3. Select the name of the connection factory, queue connection factory, or topic connection factory that you want to work with.
4. In the content pane under Additional Properties, click **Client transport properties** to view a list of the client transport settings of the WebSphere MQ connection factory, queue connection factory, or topic connection factory.

Under General Properties there are two groups of properties:
- Additional SSL settings (for more information about setting the SSL properties for WebSphere MQ, see *SSL properties* in the *Using Java* section of the WebSphere MQ information center)
- Channel exits

Make any required changes to these groups and then click **Apply** to return to the connection factory, queue connection factory or topic connection factory.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the *Using Java* section of the WebSphere MQ information center.

A WebSphere MQ connection factory, queue connection factory, or topic connection factory has the following client transport settings properties:

Certificate revocation list:

A list of LDAP URLs pointing to LDAP repositories of SSL certificates that might have been revoked.

Data type	String
Default	No certificate revocation list
Range	The value must be a space-separated list of LDAP URLs.

Peer name:

A name (possibly including wildcards) that must match the distinguished name of the peer SSL certificate for a connection to be established.

Data type	String
Default	Do not check the distinguished name of the peer certificate.
Range	Validated according to the rules for a WebSphere MQ SSLPEER channel parameter.

Reset count:

The total number of bytes to transfer over an SSL connection before renegotiating the symmetric encryption keys used to secure the connection.

Data type	Integer
Default	0 (do not renegotiate)
Range	The value must be in the range 0 through 999,999,999 (inclusive).

Receive exits:

A comma-separated list of Java class names corresponding to receive exits to be loaded.

Data type	String

Receive exit initialization data:

Initialization data to be passed to the receive exit.

Data type	String

Send exits:

A comma-separated list of Java class names corresponding to send exits to be loaded.

Data type	String

Send exit initialization data:

Initialization data to be passed to the send exit.

Data type	String

Security exit:

A Java class name corresponding to the security exit to be loaded.

Data type String

Security exit initialization data:

Initialization data to be passed to the security exit.

Data type String

Connection pool settings:

Use this page to configure connection pool settings.

This administrative console page is common to JDBC data sources and JMS connection factories (unified, queue, or topic connection factories). To view this page, the path depends on the type of resource, but generally you select an instance of the resource type then click **Connection Pool**. For example:

- Click **Resources** > **JDBC** > **Data Sources** > ***data_source*** > **[Additional Properties] Connection pool properties**
- Click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Connection pool**

Note: Connection pooling is not supported in an application client. The application client calls the database directly and does not go through a data source. If you want to use the getConnection() request from the application client, configure the JDBC provider in the application client deployment descriptors, using Rational Application Developer or an assembly tool. The connection is established between application client and the database. Application clients do not have a connection pool, but you can configure JDBC provider settings in the client deployment descriptors.

Connection timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

This value indicates the number of seconds that a connection request waits when there are no connections available in the free pool and no new connections can be created. This usually occurs because the maximum value of connections in the particular connection pool has been reached.

For example, if Connection timeout is set to 300, and the maximum number of connections are all in use, the pool manager waits for 300 seconds for a physical connection to become available. If a physical connection is not available within this time, the pool manager initiates a `ConnectionWaitTimeout` exception. In most cases, you should not retry the getConnection() method; if a longer wait time is required you should increase the Connection timeout setting value. If a `ConnectionWaitTimeout` exception is caught by the application, review the expected connection pool usage of the application and tune the connection pool and database accordingly.

If the Connection timeout is set to 0, the pool manager waits as long as necessary until a connection becomes available. This happens when the application completes a transaction and returns a connection to the pool, or when the number of connections falls below the value of Maximum Connections, and a new physical connection is created.

If Maximum Connections is set to 0, an infinite number of physical connections are enabled, and the Connection timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Maximum connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. When this number is reached, no new physical connections are created. The requester waits until a physical connection that is currently in use returns to the pool, or until a ConnectionWaitTimeoutException error displays. For example, if the Max Connections value is set to 5, and there are 5 physical connections in use, the pool manager waits for the amount of time specified in Connection timeout for a physical connection to become free.

Knowing the number of connection pools that can potentially request connections from the backend, such as a DB2 database or a CICS server, helps you determine a value for the Maximum Connections property.

For multiple stand-alone application servers that use the same data source configuration, or J2C connection factory configuration, a separate physical connection pool exists for each server. If you clone these same application servers, WebSphere Application Server (base) implements a separate connection pool for each clone.

All of these connection pools correspond to the same data source or connection factory configuration. Therefore all of these connection pools can potentially request connections from the same backend resource, at the same time. The single Maximum Connections value that you set on this console panel applies to every one of these connection pools. Consequently, setting a high Maximum Connections value can result in a load of connection requests that overwhelms your backend resource.

Data type	Integer
Default	10
Range	0 to maximum integer If Max Connections is set to 0, the Connection timeout value is ignored.

Tip: For better performance, set the value for the connection pool lower than the value for the maximum thread pool connections of the web container. To configure this setting click **Servers** > **Server types** > **WebSphere application servers** > ***server*** > **Thread Pools**, and modify the web container property. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the processor load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Minimum connections:

Specifies the minimum number of physical connections to maintain.

If the size of the connection pool is at or below the minimum connection pool size, the Unused timeout thread does not discard physical connections. However, the pool does not create connections solely to ensure that the minimum connection pool size is maintained. Also, if you set a value for Aged timeout, connections with an expired age are discarded, regardless of the minimum pool size setting.

For example, if the Minimum Connections value is set to 3, and one physical connection is created, the Unused timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the Minimum Connections setting.

Data type	Integer
Default	1
Range	0 to max int

Reap time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if Reap Time is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the Unused timeout and Aged timeout settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused timeout and Aged timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused timeout, until it reaches the number of connections specified in Minimum Connections. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread, set Reap Time to `0`, or set both Unused timeout and Aged timeout to `0`. The recommended way to disable the pool maintenance thread is to set Reap Time to `0`, and Unused timeout and Aged timeout are ignored. However, if Unused timeout and Aged timeout are set to `0`, the pool maintenance thread runs, but only physical connections that timeout, because of non-zero timeout values, are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused timeout value higher than the Reap timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections exceeds the Minimum Connections setting. For example, if the unused timeout value is set to `120`, and the pool maintenance thread is enabled (Reap Time is not `0`), any physical connection that remains unused for 2 minutes is discarded.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting Aged timeout to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged timeout value higher than the Reap timeout value for optimal performance.

For example, if the Aged timeout value is set to `1200`, and the Reap Time value is not `0`, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. The only exception is if the connection is involved in a transaction when the aged timeout is reached, the application server will not discard the connection until after the transaction is completed and the connection is closed.

The accuracy and performance of this timeout are affected by the Reap Time value. See “Reap time” on page 207 for more information.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**.

Data type	String
Defaults	• EntirePool for J2C connection factories and JMS-related connection factories • EntirePool for WebSphere Version 4.0 data sources • EntirePool for current version data sources that you create through the administrative console • EntirePool for current version data sources that you script through wsadmin AdminConfig commands, starting JDBC templates that are built into WebSphere Application Server. For information about the command createUsingTemplate, see the topic, Commands for the AdminConfig object. • FailingConnectionOnly for data sources that you script in wsadmin tool without starting JDBC templates :

Range

EntirePool
All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and issues a stale connection Exception during the next operation on that connection. Subsequent getConnection() requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this closure is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice.

FailingConnectionOnly
Only the connection that caused the stale connection exception is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a possibility that the next getConnection() request from the application can return a connection from the pool that is also stale. The result is more stale connection exceptions.

The connection pretest function attempts to insulate an application from pooled connections that are not valid. When a backend resource, such as a database, goes down, pooled connections that are not valid might exist in the free pool. This is especially true when the purge policy is failingConnectionOnly; in this case, the failing connection is removed from the pool. Depending on the failure, the remaining connections in the pool might not be valid.

Session pool settings:

Use this page to configure session pool settings.

This administrative console page is common to JMS unified connection factories, queue connection factories and topic connection factories. To view this page, you select an instance of the resource type then click **Session pools**. For example, click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Session pools**.

Connection Timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

The wait is necessary when the maximum value of connections (**Max Connections**) to a particular connection pool is reached . For example, if *Connection Timeout* is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is *not* available within this time, the Pool Manager throws a ConnectionWaitTimeoutException. It usually does not make sense to retry the getConnection() method, because if a longer wait time is required, you should set the **Connection Timeout** setting to a higher

value. Therefore, if this exception is caught by the application, the administrator should review the expected usage of the application and tune the connection pool and the database accordingly.

If Connection Timeout is set to 0, the Pool Manager waits as long as necessary until a connection is allocated (which happens when the number of connections falls below the value of **Max Connections**).

If Max Connections is set to 0, which enables an infinite number of physical connections, then the Connection Timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Max Connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use returns to the pool, or a ConnectionWaitTimeoutException is thrown.

For example, if the Max Connections value is set to 5, and there are five physical connections in use, the pool manager waits for the amount of time specified in Connection Timeout for a physical connection to become free.

If Max Connections is set to 0, the Connection Timeout value is ignored.

For better performance, set the value for the connection pool lower than the value for the Max Connections option in the web container. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

If clones are used, one data pool exists for each clone. Knowing the number of data pools is important when configuring the database maximum connections.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the CPU load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Data type	Integer
Default	10
Range	0 to max int

Min Connections:

Specifies the minimum number of physical connections to maintain.

Until this number is reached, the pool maintenance thread does not discard physical connections. However, no attempt is made to bring the number of connections up to this number. If you set a value for Aged Timeout, the minimum is not maintained. All connections with an expired age are discarded.

For example if the **Min Connections** value is set to 3, and one physical connection is created, the Unused Timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the **Min Connections** setting.

Data type	Integer
Default	1
Range	0 to max int

Reap Time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if **Reap Time** is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in **Min Connections**. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread set Reap Time to 0, or set both Unused Timeout and Aged Timeout to 0. The recommended way to disable the pool maintenance thread is to set Reap Time to 0, in which case Unused Timeout and Aged Timeout are ignored. However, if Unused Timeout and Aged Timeout are set to 0, the pool maintenance thread runs, but only physical connections that timeout due to non-zero timeout values are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused Timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the **Min Connections** setting. For example, if the unused timeout value is set to 120, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the **Reap Time** value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged Timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting **Aged Timeout** to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged Timeout value higher than the **Reap Timeout** value for optimal performance. For example, if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that

remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, are affected by the Reap Time value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge Policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**. Java EE Connector Architecture (JCA) data sources can have either option. WebSphere Version 4.0 data sources always have a purge policy of **EntirePool**.

Data type	String
Default	FailingConnectionOnly
Range	**EntirePool** All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and throws a *StaleConnectionException* during the next operation on that connection. Subsequent *getConnection* requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice. **FailingConnectionOnly** Only the connection that caused the *StaleConnectionException* is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a good possibility that the next *getConnection* request from the application can return a connection from the pool that is also stale, resulting in more stale connection exceptions.

Configuring a queue for the WebSphere MQ messaging provider

Use this task to view or change the configuration of a JMS queue destination for point-to-point messaging with the WebSphere MQ messaging provider. This task contains an optional step for you to create a new JMS queue destination.

About this task

To view or change the configuration of a queue destination for use with the WebSphere MQ messaging provider, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS->Queues** to view existing queue destinations, with a summary of their properties.
2. Select the **Scope** corresponding to the scope of the queue destinations that you want to view or change.
3. To view or change the properties of an existing queue destination, click its name in the list. Otherwise, to create a new queue, complete the following steps:
   a. Click **New** in the content pane. Select **WebSphere MQ messaging provider** then click **OK**.
   b. Specify the following required properties.

      Name The name by which this queue destination is known for administrative purposes within WebSphere Application Server.

      JNDI name
      The JNDI name that is used to bind the queue destination into the namespace.

      Queue name
      The name of the WebSphere MQ queue to which messages are sent.

   c. Click **Apply**. This defines the queue destination to WebSphere Application Server, and enables you to browse or change additional properties.
4. Optional: Change WebSphere MQ queue settings, according to your needs.
5. Click **OK**.
6. Save any changes to the master configuration.
7. To have the changed configuration take effect, stop then restart the application server.

WebSphere MQ messaging provider queue settings

Use this panel to view or change the configuration properties of the selected queue destination for point-to-point messaging with WebSphere MQ as a messaging provider.

A WebSphere MQ queue destination is used to configure the properties of a queue for the WebSphere MQ messaging provider. Connections to the queue are created using an associated WebSphere MQ queue connection factory or connection factory.

To view WebSphere MQ queue settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Queues** to display existing queue destinations.
2. Select the name of the queue destination that you want to work with.
3. Optional: To create a new queue destination, click **New**.
4. Optional: To view or change the queue destination settings, select ts name in the list displayed.

Under General Properties there are two groups of properties:

- Administration
- WebSphere MQ Queue

Make any required changes to the Administration and WebSphere MQ Queue groups of properties and then click **Apply** to save the configuration before, in the content pane under Additional Properties, you click one of the following links:

- **Advanced properties** to display or change the advanced properties of your WebSphere MQ queue destination
- **WebSphere MQ queue connection properties** to display or change the connection properties of your WebSphere MQ queue destination
- **Custom properties** to display or change the custom properties of your WebSphere MQ queue destination

Under Related items, you can click **JAAS - J2C authentication data** to configure authentication information for use with the queue destination.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ JMS resources, see the *Using Java* and *System Administration* sections of the WebSphere MQ information center.

If WebSphere MQ functionality has been disabled at a scope that affects this WebSphere MQ messaging provider resource, then an informational message indicating that WebSphere MQ has been disabled appears. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

A queue destination for use with the WebSphere MQ messaging provider has the following properties.

Scope:

The scope assigned to the queue when it is created. The scope specifies the level to which this queue definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates which are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource at a different scope, change the scope on WebSphere MQ queue destination collection panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Provider:

The JMS provider assigned when the queue is created.

For all queues created using this panel, the provider is the WebSphere MQ messaging provider.

The provider is displayed for information only.

Data type String

Name:

The name by which the queue is known for administrative purposes within WebSphere Application Server.

Data type String

JNDI name:

The name that is used to bind the queue into the JNDI namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

Data type String

Description:

A description of the queue for administrative purposes within WebSphere Application Server.

Data type String
Default Null

Queue name:

The WebSphere MQ name for the queue that holds the messages for the JMS destination.

Data type String

Queue manager or queue-sharing group name:

The name of the WebSphere MQ queue manager or queue-sharing group that hosts the queue.

Data type String

WebSphere MQ queue connection properties:

Use this panel to specify how to connect to the queue manager that hosts the queue.

The system uses these connection properties to retrieve, display and update the queue configuration details that are shown on the **WebSphere MQ queue settings** panel.

To set of change the queue connection properties, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Queues** to display existing queue destinations.
2. Click the name of the queue destination that you want to work with.
3. To view or change the queue destination settings, click its name in the list displayed.

Under General Properties there are two groups of properties:
- Administration
- WebSphere MQ Queue

Make any required changes to the Administration and WebSphere MQ Queue groups of properties and then click **Apply** before, in the content pane under Additional Properties, you click the **WebSphere MQ queue connection properties** link to display or change the connection properties of your WebSphere MQ queue destination.

Make any required changes to the General properties and then click **Apply** before, in the content pane under Additional Properties, you click **WebSphere MQ configuration** to return to the **WebSphere MQ queue settings** panel.

A queue destination for use with the WebSphere MQ messaging provider has the following WebSphere MQ queue connection properties.

Note:

- The property values that you specify must match the values that you specified when you configured the JMS resources in WebSphere MQ. For more information about configuring JMS resources in WebSphere MQ, see the *Using Java* section of the WebSphere MQ information center.
- In WebSphere MQ, names can have a maximum of 48 characters, with the exception of channels which have a maximum of 20 characters.

Note: You cannot use this panel when WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Queue Manager Host:

The name of host for the queue manager on which the queue destination is created.

Data type	String

Queue Manager Port:

The number of the port used by the queue manager on which this queue is defined.

Data type	Integer
Range	A valid TCP/IP port number. This port must be configured on the WebSphere MQ queue manager.
Default	1414

Server Connection Channel Name:

The name of the channel used for connection to the WebSphere MQ queue manager.

Data type	String
Range	1 through 20 ASCII characters

User ID:

The user ID used, with the **Password** property, for authentication when connecting to the queue manager to define the queue destination.

*If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.*

Data type	String

Password:

The password, used with the **User ID** property, for authentication when connecting to the queue manager to define the queue destination.

*If you specify a value for the **User ID** property, you must also specify a value for the **Password** property.*

Data type	String

WebSphere MQ messaging provider queue and topic advanced properties settings:

Use this panel to view or change the advanced properties for the selected queue or topic destination for use with the WebSphere MQ messaging provider. These advanced properties control the behavior of connections made to WebSphere MQ messaging provider destinations.

To view WebSphere MQ queue or topic advanced properties settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS**.
2. Click **Queues** or **Topics** to display existing queue or topic destinations.
3. If appropriate, in the content pane, change the **Scope** setting to the level at which the queue or topic destinations are defined. This restricts the set of queue or topic destinations displayed.
4. Click the name of the queue or topic destination that you want to work with.
5. In the content pane, under Additional properties, click **Advanced properties** to display a list of the advanced properties of the WebSphere MQ queue or topic destination.

Under General Properties there are five groups of properties:
- Delivery
- Message format
- Optimizations
- Message descriptor
- Additional

Make any required changes to these groups and then click **Apply** to return to the queue or topic.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring JMS resources for WebSphere MQ. For more information about configuring JMS resources for WebSphere MQ, see *Using Java* section of the WebSphere MQ information center.

A queue or topic for use with the WebSphere MQ messaging provider has the following advanced properties.

Persistence:

The level of persistence used to store messages sent to this destination.

Data type Drop-down list

Default As set by application

Range

As set by application
Messages on the destination have their persistence defined by the application that put them onto the queue.

As per WebSphere MQ queue definition
Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.

WebSphere MQ Persistent
Messages on the destination are persistent.

WebSphere MQ Non-persistent
Messages on the destination are not persistent.

WebSphere MQ High
Permit persistent messages to be sent as non-persistent messages when you use an underlying WebSphere MQ queue with a NPMCLASS of 'HIGH'.

Priority:

The priority assigned to messages sent to this destination.

Data type Drop-down list
Default As set by application
Range
As set by application
The priority of messages on this destination is defined by the application that put them onto the destination.
As per WebSphere MQ queue definition
Messages on the destination have their persistence defined by the WebSphere MQ destination definition properties.
Specified
The priority of messages on this destination is defined by the **Specified priority** property. If you select this option, you must define a priority on the **Specified priority** property.

Specified priority:

If the **Priority** property was set to Specified, select the priority assigned to messages sent to this queue type destination.

Data type Drop-down list
Units Message priority level
Default As set by application
Range 0 (lowest priority) through 9 (highest priority)

Expiry:

An option that specifies the expiry timeout for this destination.

Data type Drop-down list
Default As set by application
Range
As set by application
The expiry timeout for messages on this destination is defined by the application that put them onto the destination.
Specified
The expiry timeout for messages on this destination is defined by the **Specified expiry** property. If you select this option, you must define a timeout on the **Specified expiry** property.
Unlimited
Messages on this destination have no expiry timeout, so these messages never expire.

Specified expiry:

If the **Expiry** property is set to Specified, enter the number of milliseconds after which messages expire and are removed from this destination.

Data type Integer
Units Milliseconds
Default 0
Range Greater than or equal to 0
- 0 indicates that messages never timeout
- Other values are an integer number of milliseconds

Coded character set identifier:

The character set to use when encoding strings in the message.

Data type Integer
Default 1208
Range 1 through 65535. The coded character set identifier (CCSID) must be one of the CCSIDs supported by WebSphere MQ. Blank. Leaving this field empty indicates that the default value must be used.

For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *System Administration Guide* and *Application Programming Reference* sections of the WebSphere MQ information center.

Use native encoding:

An option that specifies whether the destination should use native encoding to provide appropriate encoding values for the Java platform.

Data type Check box
Default Selected
Range
Selected
Native encoding is used.
Cleared
Native encoding is not used, so specify the properties for **Integer encoding**, **Decimal encoding**, and **Floating point encoding**.

Integer encoding:

If the **Use native encoding** check box is cleared, select the type of integer encoding to be used.

Data type Drop-down list
Default Normal
Range
Normal
Normal integer encoding is used.
Reversed
Reversed integer encoding is used.

Decimal encoding:

If the **Use native encoding** check box is cleared, select the type of decimal encoding to be used.

Data type Drop-down list
Units Not applicable
Default Normal

Range

Normal
Normal decimal encoding is used.
Reversed
Reversed decimal encoding is used.

Floating point encoding:

If the **Use native encoding** check box is cleared, select the type of floating point encoding to be used.

Data type Drop-down list
Default IEEENORMAL
Range

IEEENORMAL
IEEE normal floating point encoding is used.
IEEEREVERSED
IEEE reversed floating point encoding is used.
z/OS z/OS floating point encoding is used.

Append RFH version 2 headers to messages sent to this destination:

The action to take when replying to a message that is sent to this destination.

Data type Check box
Default Selected
Range

Cleared
Do not append RFH version 2 headers to messages sent to this destination.
Selected
Append RFH version 2 headers to messages sent to this destination.

Message body:

Specifies whether an application processes the RFH version 2 header of a WebSphere MQ message as part of the JMS message body.

Data type Drop-down list
Default UNSPECIFIED

Range

UNSPECIFIED
When sending messages, the WebSphere MQ messaging provider does or does not generate and include an RFH version 2 header, depending on the value of the Append RFH version 2 headers to messages sent to this destination property. When receiving messages, the WebSphere MQ messaging provider acts as if the value is set to JMS.

JMS When sending messages, the WebSphere MQ messaging provider automatically generates an RFH version 2 header and includes it in the WebSphere MQ message. When receiving messages, the WebSphere MQ messaging provider sets the JMS message properties according to values in the RFH version 2 header (if these value are present); it does not present the RFH version 2 header as part of the JMS message body.

MQ When sending messages, the WebSphere MQ messaging provider does not generate an RFH version 2 header. When receiving messages, the WebSphere MQ messaging provider presents the RFH version 2 header as part of the JMS message body.

ReplyTo destination style:

Specifies the format of the JMSReplyTo field.

Data type Drop-down list

Default DEFAULT

Range

DEFAULT
The default value is equivalent to the information in the RFH version 2 header.

MQMD Use the value supplied in the MQMD. This populates the reply to queue manager field with the value from the MQMD, equivalent to the default behaviour of WebSphere MQ Version 6.0.2.4 and 6.0.2.5.

RFH2 Use the value supplied in the RFH version 2 header. If the sending application set a JMSReplyTo value, then that value is used.

Asynchronously send messages to the queue manager:

An option that enables the queue manager to acknowledge receipt of messages sent to it. Asynchronously sending messages to the queue manager is faster, but messages can be lost if the messaging infrastructure fails.

Data type Drop-down list

Default

The default value depends on whether you are working with a queue or topic destination.

As per queue definition
The default value if you are working with a queue destination.

As per topic definition
The default value if you are working with a topic destination.

Range

As per queue definition
Messages are acknowledged according to the WebSphere MQ queue definition properties.

As per topic definition
Messages are acknowledged according to the WebSphere MQ topic definition properties.

Yes The queue manager acknowledges receipt of messages sent to it.

No The queue manager does not acknowledge receipt of messages sent to it.

Read ahead and cache non-persistent messages for consumers:

An option that determines whether messages for non-persistent consumers are sent to the client speculatively. Selecting this option results in faster message delivery but messages can be lost in the event of a failure in the messaging infrastructure.

Data type Drop-down list

Default

The default value depends on whether you are working with a queue or topic destination.

As per queue definition
This is the default value if you are working with a queue destination.

As per topic definition
This is the default value if you are working with a topic destination.

Range

As per queue definition
Messages are sent to the client according to the WebSphere MQ queue definition properties.

As per topic definition
Messages are sent to the client according to the WebSphere MQ topic definition properties.

Yes Messages are sent to the client speculatively.

No Messages are not sent to the client speculatively.

ReplyTo destination style:

Specifies the format of the JMSReplyTo field.

Data type Drop-down list

Default DEFAULT

Range

DEFAULT
The default value is equivalent to the information in the RFH version 2 header.

MQMD Use the value supplied in the MQMD. This populates the reply to queue manager field with the value from the MQMD, equivalent to the default behaviour of WebSphere MQ Version 6.0.2.4 and 6.0.2.5.

RFH2 Use the value supplied in the RFH version 2 header. If the sending application set a JMSReplyTo value, then that value is used.

For more information about automatic client reconnection, see the WebSphere MQ information center.

Read ahead consumer close method:

If **Read ahead and cache non-persistent messages for consumers** is set to `Yes` or `As per queue definition` this property is enabled. This property determines what happens to messages in the internal read ahead buffer when the message consumer is closed.

Data type Drop-down list

Default Close method waits for all cached messages to be delivered

Range

Wait for all cached messages to be delivered
All messages in the internal read ahead buffer are delivered to the application's message listener before returning.

Wait for the current message to be delivered
Only the current message listener invocation completes before returning, potentially leaving messages in the internal read ahead buffer, which are then discarded.

MQMD read enabled:

Specifies whether an application can read the values of MQMD fields from JMS messages that have been sent or received using the WebSphere MQ messaging provider.

Data type Check box

Default Cleared

Range

Cleared
Applications cannot read the values of the MQMD fields.

Selected
Applications can read the values of the MQMD fields.

MQMD write enabled:

Specifies whether an application can write the values of MQMD fields to JMS messages that will be sent or received using the WebSphere MQ messaging provider.

Data type Check box

Default Cleared

Range

Cleared
Applications cannot write the values of the MQMD fields.

Selected
Applications can write the values of the MQMD fields.

MQMD message context:

Defines the message context options specified when sending messages to a destination.

Data type Drop-down list

Default DEFAULT

Range

DEFAULT
The MQOPEN API call and the MQPMO structure specify no explicit message context options.

SET_IDENTITY_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_IDENTITY_CONTEXT, and the MQPMO structure specifies MQPMO_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_ALL_CONTEXT, and the MQPMO structure specifies MQPMO_SET_ALL_CONTEXT.

Configuring a topic for the WebSphere MQ messaging provider

Use this task to view or change the configuration of a JMS topic destination for publish/subscribe messaging with the WebSphere MQ messaging provider. This task contains an optional step for you to create a new JMS topic destination.

About this task

To view or change the configuration of a topic destination for use with the WebSphere MQ messaging provider, or to create anew topic destination, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Resources** > **JMS->Topics** to view existing topic destinations, with a summary of their properties.
2. Select the **Scope** corresponding to the scope of the topic destinations that you want to view or change.
3. To view or change the properties of an existing topic destination, click its name in the list. Otherwise, to create a new topic destination, complete the following steps:
   a. Click **New** in the content pane. Select **WebSphere MQ messaging provider** then click **OK**.
   b. Specify the following required properties.

      Name The name by which this topic destination is known for administrative purposes within WebSphere Application Server.

JNDI name
The Java Naming and Directory Interface (JNDI) name that is used to bind the topic destination into the namespace.

Topic name
The name of the WebSphere MQtopic to which messages are sent.

c. Click **Apply**. This defines the topic destination to WebSphere Application Server, and enables you to browse or change additional properties.

4. Optional: Change WebSphere MQ topics settings according to your needs.
5. Click **OK**.
6. Save any changes to the master configuration.
7. To have the changed configuration take effect, stop then restart the application server.

WebSphere MQ messaging provider topic settings

Use this panel to view or change the configuration properties of the selected JMS topic destination for publish/subscribe messaging with WebSphere MQ as a messaging provider.

A WebSphere MQ topic destination is used to configure the properties of a topic for the WebSphere MQ messaging provider. Connections to the topic are created using an associated WebSphere MQ topic connection factory or connection factory.

To view WebSphere MQ topic settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->Topics** to display existing topic destinations.
2. Select the name of the topic destination that you want to work with.
3. Optional: To create a new topic destination, click **New**.
4. Optional: To view or change the topic destination settings, select its name in the list displayed.

Under General Properties there are two groups of properties:

- Administration
- WebSphere MQ topic

Make any required changes to the Administration and WebSphere MQ topic groups of properties, and then click **Apply** to save the configuration before, in the content pane under Additional Properties, you click either of the following links:

- **Advanced properties** to display or change the advanced properties of your WebSphere MQ topic.
- **Custom properties** to display or change the custom properties of your WebSphere MQ topic.

Under Related items, you can click **JAAS - J2C authentication data** to configure authentication information for use with the topic.

You can specify the following additional properties by using WebSphere MQ administrative commands:

- **wildcardFormat**
- **brokerVersion**

For more information about these properties, refer to the “createWMQTopic command” on page 904.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ JMS resources, see the *Using Java* and *MQ System Administration* sections of the WebSphere MQ information center.

If WebSphere MQ functionality has been disabled at a scope that affects this WebSphere MQ messaging provider resource, then an informational message indicating that WebSphere MQ has been disabled appears. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Scope:

The scope assigned to the topic when it is created. The scope specifies the level to which this topic definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource at a different scope, change the scope on the WebSphere MQ topic destination collection panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Provider:

The JMS provider assigned when the topic is created.

For all topics created using this panel, the provider is the WebSphere MQ messaging provider.

The provider is displayed for information only.

Data type String

Name:

The name by which the topic is known for administrative purposes within WebSphere Application Server.

Data type String

JNDI name:

The name that is used to bind the topic into the JNDI namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

Data type String

Description:

A description of the topic for administrative purposes within WebSphere Application Server.

Data type String

Topic name:

The WebSphere MQ name for the topic.

Data type String

Broker durable subscriber queue:

The queue to use for durable subscribers.

If this property is blank then the default WebSphere MQ value SYSTEM.JMS.D.SUBSCRIBER.QUEUE is used.

Data type String

Broker durable subscription connection consumer queue:

The queue to use for durable subscribers that use a connection consumer.

If this property is blank then the default WebSphere MQ value SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE is used.

Data type String

Broker publication queue:

The queue to use for publishing messages.

Data type Drop-down list
Default As connection
Range
As connection
The connection factory values are used.
Override connection
Enter the name of a queue in the associated text box.

Broker publication queue manager:

The queue manager that hosts the topic.

Data type String

WebSphere MQ messaging provider queue and topic advanced properties settings:

Use this panel to view or change the advanced properties for the selected queue or topic destination for use with the WebSphere MQ messaging provider. These advanced properties control the behavior of connections made to WebSphere MQ messaging provider destinations.

To view WebSphere MQ queue or topic advanced properties settings, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS**.
2. Click **Queues** or **Topics** to display existing queue or topic destinations.
3. If appropriate, in the content pane, change the **Scope** setting to the level at which the queue or topic destinations are defined. This restricts the set of queue or topic destinations displayed.
4. Click the name of the queue or topic destination that you want to work with.

5. In the content pane, under Additional properties, click **Advanced properties** to display a list of the advanced properties of the WebSphere MQ queue or topic destination.

Under General Properties there are five groups of properties:

- Delivery
- Message format
- Optimizations
- Message descriptor
- Additional

Make any required changes to these groups and then click **Apply** to return to the queue or topic.

Note: When specifying WebSphere MQ properties, the following restrictions apply:
- Names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.
- The property values that you specify must match the values that you specified when configuring JMS resources for WebSphere MQ. For more information about configuring JMS resources for WebSphere MQ, see *Using Java* section of the WebSphere MQ information center.

A queue or topic for use with the WebSphere MQ messaging provider has the following advanced properties.

Persistence:

The level of persistence used to store messages sent to this destination.

Data type Drop-down list

Default As set by application

Range

As set by application
Messages on the destination have their persistence defined by the application that put them onto the queue.

As per WebSphere MQ queue definition
Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.

WebSphere MQ Persistent
Messages on the destination are persistent.

WebSphere MQ Non-persistent
Messages on the destination are not persistent.

WebSphere MQ High
Permit persistent messages to be sent as non-persistent messages when you use an underlying WebSphere MQ queue with a NPMCLASS of 'HIGH'.

Priority:

The priority assigned to messages sent to this destination.

Data type Drop-down list

Default As set by application

Range

As set by application
The priority of messages on this destination is defined by the application that put them onto the destination.

As per WebSphere MQ queue definition
Messages on the destination have their persistence defined by the WebSphere MQ destination definition properties.

Specified
The priority of messages on this destination is defined by the **Specified priority** property. If you select this option, you must define a priority on the **Specified priority** property.

Specified priority:

If the **Priority** property was set to Specified, select the priority assigned to messages sent to this queue type destination.

Data type Drop-down list
Units Message priority level
Default As set by application
Range 0 (lowest priority) through 9 (highest priority)

Expiry:

An option that specifies the expiry timeout for this destination.

Data type Drop-down list
Default As set by application
Range

As set by application
The expiry timeout for messages on this destination is defined by the application that put them onto the destination.

Specified
The expiry timeout for messages on this destination is defined by the **Specified expiry** property. If you select this option, you must define a timeout on the **Specified expiry** property.

Unlimited
Messages on this destination have no expiry timeout, so these messages never expire.

Specified expiry:

If the **Expiry** property is set to Specified, enter the number of milliseconds after which messages expire and are removed from this destination.

Data type Integer
Units Milliseconds
Default 0
Range Greater than or equal to 0
- 0 indicates that messages never timeout
- Other values are an integer number of milliseconds

Coded character set identifier:

The character set to use when encoding strings in the message.

Data type Integer
Default 1208
Range 1 through 65535. The coded character set identifier (CCSID) must be one of the CCSIDs supported by WebSphere MQ. Blank. Leaving this field empty indicates that the default value must be used.

For more information about supported CCSIDs, and about converting between message data from one coded character set to another, see the *System Administration Guide* and *Application Programming Reference* sections of the WebSphere MQ information center.

Use native encoding:

An option that specifies whether the destination should use native encoding to provide appropriate encoding values for the Java platform.

Data type Check box
Default Selected
Range
Selected
Native encoding is used.
Cleared
Native encoding is not used, so specify the properties for **Integer encoding**, **Decimal encoding**, and **Floating point encoding**.

Integer encoding:

If the **Use native encoding** check box is cleared, select the type of integer encoding to be used.

Data type Drop-down list
Default Normal
Range
Normal
Normal integer encoding is used.
Reversed
Reversed integer encoding is used.

Decimal encoding:

If the **Use native encoding** check box is cleared, select the type of decimal encoding to be used.

Data type Drop-down list
Units Not applicable
Default Normal
Range
Normal
Normal decimal encoding is used.
Reversed
Reversed decimal encoding is used.

Floating point encoding:

If the **Use native encoding** check box is cleared, select the type of floating point encoding to be used.

Data type Drop-down list
Default IEEENORMAL
Range
IEEENORMAL
IEEE normal floating point encoding is used.
IEEEREVERSED
IEEE reversed floating point encoding is used.
z/OS z/OS floating point encoding is used.

Append RFH version 2 headers to messages sent to this destination:

The action to take when replying to a message that is sent to this destination.

Data type Check box
Default Selected
Range
Cleared
Do not append RFH version 2 headers to messages sent to this destination.
Selected
Append RFH version 2 headers to messages sent to this destination.

Message body:

Specifies whether an application processes the RFH version 2 header of a WebSphere MQ message as part of the JMS message body.

Data type Drop-down list
Default UNSPECIFIED
Range
UNSPECIFIED
When sending messages, the WebSphere MQ messaging provider does or does not generate and include an RFH version 2 header, depending on the value of the Append RFH version 2 headers to messages sent to this destination property. When receiving messages, the WebSphere MQ messaging provider acts as if the value is set to JMS.
JMS When sending messages, the WebSphere MQ messaging provider automatically generates an RFH version 2 header and includes it in the WebSphere MQ message. When receiving messages, the WebSphere MQ messaging provider sets the JMS message properties according to values in the RFH version 2 header (if these value are present); it does not present the RFH version 2 header as part of the JMS message body.
MQ When sending messages, the WebSphere MQ messaging provider does not generate an RFH version 2 header. When receiving messages, the WebSphere MQ messaging provider presents the RFH version 2 header as part of the JMS message body.

ReplyTo destination style:

Specifies the format of the JMSReplyTo field.

Data type Drop-down list

Default DEFAULT

Range

DEFAULT
The default value is equivalent to the information in the RFH version 2 header.

MQMD Use the value supplied in the MQMD. This populates the reply to queue manager field with the value from the MQMD, equivalent to the default behaviour of WebSphere MQ Version 6.0.2.4 and 6.0.2.5.

RFH2 Use the value supplied in the RFH version 2 header. If the sending application set a JMSReplyTo value, then that value is used.

Asynchronously send messages to the queue manager:

An option that enables the queue manager to acknowledge receipt of messages sent to it. Asynchronously sending messages to the queue manager is faster, but messages can be lost if the messaging infrastructure fails.

Data type Drop-down list

Default The default value depends on whether you are working with a queue or topic destination.

As per queue definition
The default value if you are working with a queue destination.

As per topic definition
The default value if you are working with a topic destination.

Range

As per queue definition
Messages are acknowledged according to the WebSphere MQ queue definition properties.

As per topic definition
Messages are acknowledged according to the WebSphere MQ topic definition properties.

Yes The queue manager acknowledges receipt of messages sent to it.

No The queue manager does not acknowledge receipt of messages sent to it.

Read ahead and cache non-persistent messages for consumers:

An option that determines whether messages for non-persistent consumers are sent to the client speculatively. Selecting this option results in faster message delivery but messages can be lost in the event of a failure in the messaging infrastructure.

Data type Drop-down list

Default The default value depends on whether you are working with a queue or topic destination.

As per queue definition
This is the default value if you are working with a queue destination.

As per topic definition
This is the default value if you are working with a topic destination.

Range

As per queue definition
Messages are sent to the client according to the WebSphere MQ queue definition properties.
As per topic definition
Messages are sent to the client according to the WebSphere MQ topic definition properties.
Yes Messages are sent to the client speculatively.
No Messages are not sent to the client speculatively.

ReplyTo destination style:

Specifies the format of the JMSReplyTo field.

Data type Drop-down list
Default DEFAULT
Range

DEFAULT
The default value is equivalent to the information in the RFH version 2 header.
MQMD Use the value supplied in the MQMD. This populates the reply to queue manager field with the value from the MQMD, equivalent to the default behaviour of WebSphere MQ Version 6.0.2.4 and 6.0.2.5.
RFH2 Use the value supplied in the RFH version 2 header. If the sending application set a JMSReplyTo value, then that value is used.

For more information about automatic client reconnection, see the WebSphere MQ information center.

Read ahead consumer close method:

If **Read ahead and cache non-persistent messages for consumers** is set to `Yes` or `As per queue definition` this property is enabled. This property determines what happens to messages in the internal read ahead buffer when the message consumer is closed.

Data type Drop-down list
Default Close method waits for all cached messages to be delivered
Range

Wait for all cached messages to be delivered
All messages in the internal read ahead buffer are delivered to the application's message listener before returning.
Wait for the current message to be delivered
Only the current message listener invocation completes before returning, potentially leaving messages in the internal read ahead buffer, which are then discarded.

MQMD read enabled:

Specifies whether an application can read the values of MQMD fields from JMS messages that have been sent or received using the WebSphere MQ messaging provider.

Data type Check box
Default Cleared

Range

Cleared
Applications cannot read the values of the MQMD fields.

Selected
Applications can read the values of the MQMD fields.

MQMD write enabled:

Specifies whether an application can write the values of MQMD fields to JMS messages that will be sent or received using the WebSphere MQ messaging provider.

Data type Check box
Default Cleared
Range

Cleared
Applications cannot write the values of the MQMD fields.

Selected
Applications can write the values of the MQMD fields.

MQMD message context:

Defines the message context options specified when sending messages to a destination.

Data type Drop-down list
Default DEFAULT
Range

DEFAULT
The MQOPEN API call and the MQPMO structure specify no explicit message context options.

SET_IDENTITY_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_IDENTITY_CONTEXT, and the MQPMO structure specifies MQPMO_SET_IDENTITY_CONTEXT.

SET_ALL_CONTEXT
The MQOPEN API call specifies the message context option MQOO_SET_ALL_CONTEXT, and the MQPMO structure specifies MQPMO_SET_ALL_CONTEXT.

Configuring custom properties for WebSphere MQ messaging provider JMS resources

In addition to the standard properties that you can define for WebSphere MQ messaging provider JMS resources, you can configure further WebSphere MQ properties as custom properties. You can configure custom properties for activation specifications, connection factories, and JMS destinations for the WebSphere MQ messaging provider.

About this task

WebSphere Application Server supports the use of custom properties to define WebSphere MQ properties. This is useful because it enables WebSphere Application Server to work with later versions of WebSphere MQ that might have properties that are not exposed in WebSphere Application Server.

You can use this task to set custom properties for JMS destinations, connection factories, and activation specifications for the WebSphere MQ messaging provider. For example, you would use this task to set the custom property WAS_EndpointInitialState for an activation specification. WAS_EndpointInitialState determines whether or not message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.

To specify custom properties for WebSphere MQ messaging provider JMS resources, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS**.
2. Select the type of JMS resource for which you want create custom properties (JMS destinations, connection factories, or activation specifications).
3. Optional: Change the **Scope** setting to the level at which the resource definition is visible to applications.
4. In the contents pane, select the specific WebSphere MQ messaging provider JMS resource name for which you want to specify the custom property. This displays information about the resource.
5. To create a custom property, select **[Additional Properties] Custom properties**.
6. Click **New** in the content pane.
7. Specify the following attributes for the new custom property:

   Name
   The name of the custom property. A property name is required.

   Value The value for the custom property.

   Type The type of the custom property. Select the custom property type from the list.
8. Click **Apply**. This defines the custom property to WebSphere Application Server, and enables you to browse or change additional properties.
9. Save any changes to the master configuration.
10. To have the changed configuration take effect, stop then restart the application server.

Example

To set the custom property WAS_EndpointInitialState for an activation specification, specify the following attributes:

- For **Name**, specify `WAS_EndpointInitialState`.
- For **Value**, specify one of the following options:

  `ACTIVE` Specify a value of `ACTIVE` if message consumption is to begin as soon as the message-driven bean connects with the JMS destination.

  `INACTIVE`
  Specify a value of `INACTIVE` if message consumption is not to begin until you use the wsadmin tool or the administrative console to activate message consumption for the message-driven bean.
- For **Type**, select String.

WebSphere MQ messaging provider custom properties

WebSphere MQ messaging provider custom properties can be specified in the administrative console. Click **Resources** > **JMS**. Select the type of JMS resource for which you want create custom properties (JMS destinations, connection factories, or activation specifications). In the contents pane, select the specific WebSphere MQ messaging provider JMS resource name. Information about the resource is displayed. To create custom properties, select **[Additional Properties] Custom properties**.

You can define the following WebSphere MQ messaging provider custom properties:

- "WAS_EndpointInitialState"

WAS_EndpointInitialState

This custom property is for use with activation specifications.

This parameter determines whether the endpoint is activated when the endpoint is registered.

This parameter should be ignored for subsequent activation or deactivation via the J2CMessageEndpoint.

Table 61. WAS_EndpointInitialState custom property. The table includes the data type and acceptable values for the property.

Data type	String
Acceptable values	ACTIVE, INACTIVE

Configuring custom properties for WebSphere MQ messaging provider JMS resources

In addition to the standard properties that you can define for WebSphere MQ messaging provider JMS resources, you can configure further WebSphere MQ properties as custom properties. You can configure custom properties for activation specifications, connection factories, and JMS destinations for the WebSphere MQ messaging provider.

About this task

WebSphere Application Server supports the use of custom properties to define WebSphere MQ properties. This is useful because it enables WebSphere Application Server to work with later versions of WebSphere MQ that might have properties that are not exposed in WebSphere Application Server.

You can use this task to set custom properties for JMS destinations, connection factories, and activation specifications for the WebSphere MQ messaging provider. For example, you would use this task to set the custom property WAS_EndpointInitialState for an activation specification. WAS_EndpointInitialState determines whether or not message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.

To specify custom properties for WebSphere MQ messaging provider JMS resources, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS**.
2. Select the type of JMS resource for which you want create custom properties (JMS destinations, connection factories, or activation specifications).
3. Optional: Change the **Scope** setting to the level at which the resource definition is visible to applications.
4. In the contents pane, select the specific WebSphere MQ messaging provider JMS resource name for which you want to specify the custom property. This displays information about the resource.
5. To create a custom property, select **[Additional Properties] Custom properties**.
6. Click **New** in the content pane.
7. Specify the following attributes for the new custom property:

   Name

   The name of the custom property. A property name is required.

Value The value for the custom property.

Type The type of the custom property. Select the custom property type from the list.

8. Click **Apply**. This defines the custom property to WebSphere Application Server, and enables you to browse or change additional properties.
9. Save any changes to the master configuration.
10. To have the changed configuration take effect, stop then restart the application server.

Example

To set the custom property WAS_EndpointInitialState for an activation specification, specify the following attributes:

- For **Name**, specify `WAS_EndpointInitialState`.
- For **Value**, specify one of the following options:

  `ACTIVE` Specify a value of `ACTIVE` if message consumption is to begin as soon as the message-driven bean connects with the JMS destination.

  `INACTIVE`
  Specify a value of `INACTIVE` if message consumption is not to begin until you use the wsadmin tool or the administrative console to activate message consumption for the message-driven bean.
- For **Type**, select String.

WebSphere MQ resource custom properties settings

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

To view this administrative console page, complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. If appropriate, in the content pane, change the scope of the WebSphere MQ messaging provider.
3. In the content pane, click the **WebSphere MQ messaging provider** that you want to support the JMS destination.
4. In the content pane, under Additional Properties, click the type of resource that you want to change, for example **Queues**.
5. Click the name of the resource that you want to work with.
6. In the content pane, under General Properties, complete the groups of fields, for example **Administration** and **WebSphere MQ Queue**.
7. In the content pane, under Additional Properties, click **Custom properties** to display a list of the custom properties of a WebSphere MQ resource.

A resource for use with the WebSphere MQ messaging provider has the following custom properties.

Note:

- The property values that you specify must match the values that you specified when configuring WebSphere MQ for JMS resources. For more information about configuring WebSphere MQ for JMS resources, see the *Using Java* section of the WebSphere MQ information center.
- In WebSphere MQ, names can have a maximum of 48 characters, with the exception of channels, which have a maximum of 20 characters.

Name:

The name by which the resource is known for administrative purposes within WebSphere Application Server.

Data type String

Value:

The value of the resource.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Description:

A description of the resource, for administrative purposes within WebSphere Application Server.

Data type String
Default Null

Required:

Whether the resource is required, for administrative purposes within WebSphere Application Server.

Data type String
Default Null

Configuring properties for the WebSphere MQ resource adapter

You can configure the WebSphere MQ resource adapter properties that affect the connection pool, which is used by WebSphere MQ messaging provider activation specifications.

About this task

There are four properties that are used to configure the WebSphere MQ resource adapter used by the WebSphere MQ messaging provider:

- **maxConnections**
- **connectionConcurrency**
- **reconnectionRetryCount**
- **reconnectionRetryInterval**

These four properties affect the connection pool, which is used by the WebSphere MQ messaging provider activation specifications. They do not affect the WebSphere MQ messaging provider queues, topics, or connection factories.

For further information about these properties, see Configuration of the ResourceAdapter object in the WebSphere MQ information center.

To configure the WebSphere MQ resource adapter properties, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS->JMS providers** to display a list of JMS providers in the content pane.
2. If you want to manage JMS resources that are defined at a different scope setting, change the **Scope** setting to the required level.

   Note: These properties only have an effect on the WebSphere MQ messaging provider objects that are defined at the same scope as the messaging provider and resource adapter on which they are set. So, for example, if you set the **max connections** property to a particular setting at the server scope, only the server scoped WebSphere MQ messaging provider activation specifications are affected by this setting.
3. In the Providers column of the displayed list of JMS providers, click the name of the WebSphere MQ messaging provider that you want to work with.
4. In the content pane under Additional properties, click **Resource adapter properties** to view the configuration page for the properties.
5. Specify the required values for the properties:

   Max connections
   The maximum number of connections to a WebSphere MQ queue manager.

   Connection concurrency
   The maximum number of message-driven beans that can be supplied by each connection.

   Reconnection retry count
   The maximum number of attempts made by a WebSphere MQ messaging provider activation specification to reconnect to a WebSphere MQ queue manager if a connection fails.

   Reconnection retry interval
   The time, in milliseconds, that a WebSphere MQ messaging provider activation specification waits before making another attempt to reconnect to a WebSphere MQ queue manager.
6. Click **Apply**. This defines the property to WebSphere Application Server, and enables you to browse or change additional properties.
7. Save any changes to the master configuration.
8. To have the changed configuration take effect, stop then restart the application server.

Configuring custom properties for the WebSphere MQ resource adapter

In addition to the standard properties that you can specify for the WebSphere MQ resource adapter, you can configure custom properties for WebSphere MQ resource adapter.

About this task

WebSphere Application Server supports the definition of custom properties for the WebSphere MQ resource adapter. This is useful because it enables WebSphere Application Server to work with later versions of WebSphere MQ that might have properties that are not exposed in WebSphere Application Server.

To specify custom properties for the WebSphere MQ resource adapter, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Resources** > **JMS->JMS providers** to display a list of JMS providers in the content pane.
2. If you want to manage JMS resources that are defined at a different scope setting, change the **Scope** setting to the required level.

3. In the Providers column of the displayed list of JMS providers, click the name of the WebSphere MQ messaging provider that you want to work with.
4. In the content pane under Additional properties, click **Resource adapter properties** to view the configuration page for the properties.
5. In the content pane under Additional properties, click **Custom properties** to view the configuration page for the properties.
6. Click the name of the property that you want to configure then make any required changes.
7. Click **Apply**. This defines the custom property to WebSphere Application Server, and enables you to browse or change additional properties.
8. Save any changes to the master configuration.
9. To have the changed configuration take effect, stop then restart the application server.

Disabling WebSphere MQ functionality in WebSphere Application Server

If you do not need to use WebSphere MQ functionality in an application server you can disable it by using either the administrative console or an administrative command. You can also disable WebSphere MQ functionality in a client process by specifying a custom property.

About this task

Note: When a WebSphere Application Server process or an application client process starts, and while this process is running, an amount of processing is performed to allow it to support WebSphere MQ-related functionality such as the WebSphere MQ messaging provider. By default this processing is performed regardless of whether any WebSphere MQ-related functionality is ever used. If you do not need to take advantage of any WebSphere MQ functionality, it is possible to disable all WebSphere MQ functionality in an application server or client process to give increased performance.

Disabling WebSphere MQ functionality in a WebSphere Application Server process has the following effects:

- No WebSphere MQ messaging provider functionality is available on that particular server:
  - Any defined WebSphere MQ messaging provider resources are not bound into JNDI, and so are unavailable to look up from inside the affected application server process, from other application server processes or application clients.
  - It is still possible to define WebSphere MQ messaging provider resources. However the **test connection** button on either the create connection factory or create activation specification wizard, depending on the scope at which WebSphere MQ has been disabled, does not work.
  - Any message driven beans that use message listener ports configured with WebSphere MQ messaging provider resources do not start.
  - Any message driven beans that use WebSphere MQ messaging provider activation specifications do not start.
  - It is not possible to recover any indoubt XA transactions involving WebSphere MQ messaging provider resources.
  - Any attempt to look up a WebSphere MQ messaging provider resource from a remote server that does not have WebSphere MQ functionality disabled fails.
  - It is not possible to use the WebSphere MQ queue connection properties function.
- No WebSphere MQ link functionality is available on that particular server:
  - It is not possible to stop or start any WebSphere MQ links.
  - It is not possible to stop or start any WebSphere MQ receiver channels.
  - It is not possible to stop, start, or reset any WebSphere MQ sender channels.

  - It is not possible to send messages to a WebSphere MQ queue manager. Any messages that are sent to a foreign bus based on a WebSphere MQ link remain on the transmission item stream for that WebSphere MQ link.
  - It is not possible to receive messages from a WebSphere MQ queue manager.
  - The inbound channel chains used by the WebSphere MQ link do not start.
  - It is not possible to resolve indoubt sender channels.
  - Attempts to use the **Test connection** functionality of the foreign bus connection that uses the WebSphere MQ link fail.
  - It is not possible to fully delete a WebSphere MQ link, as any stored state about indoubt messages cannot be processed.
- No WebSphere MQ server functionality is available on that particular server:
  - It is not possible to send messages to WebSphere MQ.
  - It is not possible to receive messages from WebSphere MQ.
  - The **Test connection** button does not work.
- No WebSphere MQ client link functionality works:
  - It is not possible to stop or start any WebSphere MQ client links.
  - It is not possible to send messages using a WebSphere MQ client link.
  - It is not possible to receive messages using a WebSphere MQ client link.
  - The inbound channel chains used by the WebSphere MQ link do not start.
- WebSphere MQ resource adapters do not start.
- WebSphere MQ Base Java functionality is unavailable.
- Any attempt to use any classes provided by WebSphere MQ fails.

Disabling WebSphere MQ functionality in a WebSphere Application Server client process has the following effects:

- Any attempt to look up a WebSphere MQ messaging provider resource from a remote server that does not have WebSphere MQ functionality disabled fails.
- WebSphere MQ Base Java functionality is not available.
- Any attempt to make use of any classes provided by WebSphere MQ fails.

Procedure

- To disable WebSphere MQ functionality in a WebSphere Application Server process, complete one of the following steps:
  - Using the administrative console, select the **Disable WebSphere MQ** check box on the required WebSphere MQ messaging provider panel.
  - Use the manageWMQ administrative command with the disableWMQ flag.

  In a single server environment, you can only disable WebSphere MQ at the server scope. When you have saved your changes and restarted the application server, all WebSphere MQ functionality is disabled on that server.

  In a network deployment environment, you can disable WebSphere MQ at all scopes in order to give fine grained configuration flexibility:
  - At the cell scope, all WebSphere MQ functionality is disabled on all application servers in the cell.
  - At the node scope, all WebSphere MQ functionality is disabled on all application servers that are part of that node.
  - At the cluster scope, all WebSphere MQ functionality is disabled on all application servers in that cluster.
  - At the server scope, all WebSphere MQ functionality is disabled in that particular application server.

The value of the **Disable WebSphere MQ** check box at a higher scope takes precedence over the value at a lower scope. For example, if you do not select the check box at the server scope but do select it for a higher (for example, cell) scoped WebSphere MQ messaging provider, the value at the cell scope takes precedence and WebSphere MQ functionality is therefore disabled in all application servers in the cell, regardless of whether the check box is selected at the server scope. The changes take effect when you have saved them and restarted all affected processes in the cell.

- To disable WebSphere MQ functionality in a WebSphere Application Server client process, specify the custom property com.ibm.ejs.jms.disableWMQSupport=true.

Example

Consider the following example: A network deployment configuration with two nodes: node1 and node2. Node1 has two servers on it, server1 and server2. Node2 has a single server on it, server3. Server3 and server1 are part of a cluster, cluster1. The WebSphere MQ messaging provider panel at cluster1 scope has the **WebSphere MQ disabled** check box selected and the changes saved. When cluster1 has been restarted, all WebSphere MQ functionality is disabled on server3 and server1.

It is worth noting that it is possible to have WebSphere MQ functionality disabled on all processes in a network deployment configuration without all scopes having WebSphere MQ functionality disabled. Using the scenario in the previous example, if all nodes in the topology (deployment manager node, node1 and node2) have WebSphere MQ functionality disabled, then all the processes in the topology also have WebSphere MQ functionality disabled.

What to do next

When the server starts, it is possible to detect whether WebSphere MQ functionality has been disabled on that server because messages with the following ids are output:

- `WMSG2016I` is output when the server starts if WebSphere MQ has been disabled.
- `CWSIC3650I` is output once for any configured WebSphere MQ links that are running on the server.
- `CWSIC3713I` is output once for any configured WebSphere MQ client links that are running on the server.

WMQAdminCommands command group for the AdminTask object

You can use the WebSphere MQ administrative commands to manage JMS resources for the WebSphere MQ messaging provider.

You can configure JMS resources for the WebSphere MQ messaging provider through the WebSphere MQ administrative commands, or you can configure JMS resources for the WebSphere MQ messaging provider through the administrative console.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

These commands are valid only when they are used with WebSphere Application Server Version 7 and later application servers. Do not use them with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using these commands, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

The following commands are available for the WMQAdminCommands group of the AdminTask object:

- createWMQActivationSpec command
- deleteWMQActivationSpec command
- listWMQActivationSpec command
- modifyWMQActivationSpec command
- showWMQActivationSpec command
- createWMQConnectionFactory command
- deleteWMQConnectionFactory command
- listWMQConnectionFactory command
- modifyWMQConnectionFactory command
- showWMQConnectionFactory command
- createWMQTopic command
- deleteWMQTopic command
- listWMQTopic command
- modifyWMQTopic command
- showWMQTopic command
- manageWMQ command
- migrateWMQMLP command
- createWMQQueue command
- deleteWMQQueue command
- listWMQQueue command
- modifyWMQQueue command
- showWMQQueue command

createWMQActivationSpec command

Use the createWMQActivationSpec command to create an activation specification for the WebSphere MQ messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the createWMQActivationSpec command to create a WebSphere MQ messaging provider activation specification at a specific scope.

You cannot create a WebSphere MQ messaging provider activation specification under either of the following conditions:

- A WebSphere MQ messaging provider activation specification already exists with the same name, at the same scope.
- The JNDI name clashes with another entry in WebSphere Application Server JNDI.

You create a CCDT based activation specification by specifying any of the following parameters:

- **-ccdtUrl**
- **-ccdtQmgrName**

If you do not specify any of the following parameters, you create a generic activation specification :

- **-ccdtUrl**
- **-ccdtQmgrName**

Target object

The scope of the WebSphere MQ messaging provider at which the WebSphere MQ messaging provider activation specification is to be created.

Required parameters

-name
The administrative name assigned to this WebSphere MQ messaging provider activation specification.

-jndiName
The name and location used to bind this object into WebSphere Application Server JNDI.

-destinationJndiName
The JNDI name of a WebSphere MQ messaging provider queue or topic type destination. When an MDB is deployed with this activation specification, messages for the MDB are consumed from this destination.

-destinationType
The type of the destination specified by using the **-destinationJndiName** parameter.

Enter one of the following values:

- javax.jms.Queue
- javax.jms.Topic

There is no default value.

Optional parameters

-description
An administrative description assigned to the activation specification.

-ccdtUrl
A URL to a client channel definition table to use, for this activation specification, when contacting WebSphere MQ.

Use this parameter to create ccdtURL activation specifications

Do not specify this parameter in conjunction with the following parameters: **-qmgrName**, **-qmgrType**, **-qmgrHostname**, **-qmgrPortNumber**, **-qmgrSvrconnChannel**, or **-localAddress**.

-ccdtQmgrName
A queue manager name, used to select one or more entries from a client channel definition table.

You must specify this parameter if the **-transportType** has been specified as client or bindingsThenClient.

Do not specify this parameter in conjunction with the following parameters: **-qmgrName**, **-qmgrType**, **-qmgrHostname**, **-qmgrPortNumber**, **-qmgrSvrconnChannel**, or **-localAddress**.

-qmgrName
The name of the queue manager to use, for this activation specification, when contacting WebSphere MQ.

Use this parameter to create generic activation specifications.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

-wmqTransportType
This parameter determines the way in which a connection is established to WebSphere MQ for this activation specification.

Use this parameter to create generic activation specifications.

Enter one of the following case-sensitive values:

- BINDINGS
- BINDINGS_THEN_CLIENT
- CLIENT

BINDINGS_THEN_CLIENT is the default value.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

For more information about configuring a transport type of bindings then client or bindings, refer to “Configuring the WebSphere MQ messaging provider with native libraries information” on page 723.

-qmgrHostname
The host name to use, for this activation specification, when attempting a client mode connection to WebSphere MQ. It must be a valid TCP/IP host name or IPv4 or IPv6 address.

The default value is the local host.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

For information on setting the **-qmgrHostname** parameter in conjunction with the **-connectionNameList** parameter, see the description of the **-connectionNameList** parameter.

-qmgrPortNumber
The port number to use, for this activation specification, when attempting a client mode connection to WebSphere MQ.

Enter an integer value in the range 1 - 65536 (inclusive).

The default value is 1414.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

For information on setting the **-qmgrPortNumber** parameter in conjunction with the **-connectionNameList** parameter, see the description of the **-connectionNameList** parameter.

-connectionNameList

The connection name list specifying the host name and port details to use when you want the activation specification to connect to a multi-instance queue manager.

Note: You must only use the **-connectionNameList** parameter to allow a connection to a multi-instance queue manager. Using the **-connectionNameList** parameter to connect to a non-multi-instance queue manager can jeopardize transaction integrity.

The **-connectionNameList** parameter must be entered as a comma separated list of host names and ports in the following format:

host(*port*),*host*(*port*)

For *host* enter a valid TCP/IP host name, IPv4 or IPv6 address.

For *port* enter an integer value between 1 and 65536 (inclusive). Specifying a value for *port* is optional. When you do not specify a value, *port* defaults to 1414.

For example: `localhost(1234),remotehost(1234),remotehost2`

When you specify the **-connectionNameList** parameter, the **-qmgrHostname** and **-qmgrPortNumber** parameters are automatically set to the host name and port number of the first entry in the connection name list. This overrides any values that you previously specified in the **-qmgrHostname** and **-qmgrPortNumber** parameters.

In the preceding example this would mean that **-qmgrHostname** would be `localhost` and **-qmgrPortNumber** would be `1234`.

The **-connectionNameList** parameter is only valid for use in WebSphere Application Server Version 8.0.

Attempting to specify the **-connectionNameList** parameter on a WebSphere MQ messaging provider activation specification which is defined at a server or node scope that is running on a version of WebSphere Application Server earlier than Version 8.0 results in an error message when you run the following commands:

- createWMQActivationSpec
- modifyWMQActivationSpec

If you specify the **-connectionNameList** parameter on a cell or cluster-scoped WebSphere MQ activation specification, you can use it for nodes that are running WebSphere Application Server Version 7.0. The exact behaviour is determined by the fix pack level of the node:

- For nodes running at WebSphere Application Server Version 7.0 Fix Pack 7 or later, the activation specification uses the **-connectionNameList** parameter to connect to a multi-instance queue manager.
- For nodes running at a fix pack level earlier than WebSphere Application Server Version 7.0 Fix Pack 7, a warning message similar to the one in the following example is output:

```
[29/09/10 12:15:27:468 BST] 00000018 J2CUtilityCla W
J2CA0008W: Class com.ibm.mq.connector.inbound.ActivationSpecImpl used by resource
cells/L3A3316Node01Cell/resources.xml#J2CResourceAdapter_1284547647859 did not contain
method setConnectionNameList. Processing continued.
```

You can ignore this message.

You must not specify the **-connectionNameList** parameter in conjunction with the **-ccdtUrl** or **-ccdtQmgrName** parameters.

Note: If you use the**-connectionNameList** parameter with a centrally managed SSL configuration the host name and port number information used to select the appropriate SSL configuration is based on the first entry in the **-connectionNameList**, regardless of which entry in the list is actually used to connect to the queue manager. This is because each instance of a multi-instance queue manager should be using the same SSL configuration, for a given server connection channel, regardless of which instance is actually running.

For more information on using multi-instance queue managers, see the WebSphere MQ information center.

-authAlias
The authentication alias used to obtain the credentials specified when this activation specification needs to establish a connection to WebSphere MQ.

-clientId
The client identifier used for connections started by using this activation specification.

-providerVersion
This parameter determines the minimum version, and capabilities of the queue manager.

Enter values in one of the following formats:

- *n*
- *n.n*
- *n.n.n*
- *n.n.n.n*

where *n* is an integer greater than or equal to zero.

For example 6.0.0.0

-sslCrl
This parameter specifies a list of LDAP servers that are used to provide certificate revocation information if this activation specification establishes an SSL based connection to WebSphere MQ.

-sslResetCount
This parameter is used when the activation specification establishes an SSL connection to the queue manager. This parameter determines how many bytes to transfer before resetting the symmetric encryption key that is used for the SSL session.

Enter a value in the range 0 through 999,999,999.

The default value is 0.

-sslPeerName
This parameter is used when the activation specification establishes an SSL connection to the queue manager. The value is compared with the distinguished name present in the peer's certificate.

-rcvExit
A comma-separated list of receive exit class names.

-rcvExitInitData
Initialization data to pass to the receive exit.

Do not specify this parameter unless you specify the **-rcvExit** parameter.

-sendExit
A comma-separated list of send exit class names.

-sendExitInitData
Initialization data to pass to the send exit.

Do not specify this parameter unless you specify the **-sendExit** parameter.

-secExit
A security exit class name.

-secExitInitData
Initialization data to pass to the security exit.

Do not specify this parameter unless you specify the **-secExit** parameter.

-compressHeaders
This parameter determines if message headers are compressed.

Enter one of the following values:

- NONE
- SYSTEM

The default value is NONE.

-compressPayload
This parameter determines if message payloads are compressed.

Enter one of the following values:

- NONE
- RLE
- ZLIBFAST
- ZLIBHIGH

The default value is NONE.

-msgRetention
This parameter determines if the connection consumer keeps unwanted messages on the input queue.

Enter one of the following values:

- YES
- NO

where YES specifies that the connection consumer keeps unwanted messages on the input queue, and NO specifies that the messages are disposed of according to their disposition options.

The default value is YES.

-rescanInterval
When a message consumer in the point-to-point domain uses a message selector to select which messages it is to receive, the JMS client searches the WebSphere MQ queue for suitable messages in the sequence determined by the MsgDeliverySequence attribute of the queue. When the client finds a suitable message and delivers it to the consumer, the client resumes the search for the next suitable message from its current position in the queue. The client continues to search the queue in this way until it reaches the end of the queue, or until the interval of time in milliseconds, as determined by the value of this **-rescanInterval** parameter has expired. In each case, the client returns to the beginning of the queue to continue its search, and a new time interval commences.

This parameter must be a positive integer value.

The default value is 5000.

-ccsid
The coded character set identifier (CCSID) to be used on connections.

The value of this parameter must be a positive integer and must be one of the CCSIDs supported by WebSphere MQ. See the “WebSphere MQ messaging provider activation specification advanced properties” on page 749 for more details.

The default value is 819.

-failIfQuiescing
This parameter determines the behavior of certain calls to the queue manager when the queue manager is put into quiescing state.

The value of this parameter must be true or false.

true specifies that calls to certain methods fail if the queue manager is in a quiescing state. If an application detects that the queue manager is quiescing, the application can complete its immediate task and close the connection, allowing the queue manager to stop.

false specifies that no methods fail if the queue manager is in a quiescing state. If you specify this value, an application cannot detect that the queue manager is quiescing. The application might continue to perform operations against the queue manager, and therefore prevent the queue manager from stopping.

The default value is true.

-brokerCtrlQueue
The name of the broker control queue to use if this activation specification is to subscribe to a topic.

The default value is SYSTEM.BROKER.CONTROL.QUEUE.

-brokerSubQueue
The name of the queue to use for obtaining subscription messages if this activation specification is to subscribe to a topic.

The default value is SYSTEM.JMS.ND.SUBSCRIBER.QUEUE.

-brokerCCSubQueue
The name of the queue from which non-durable subscription messages are retrieved for a ConnectionConsumer.

The default value is SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE.

-brokerVersion
The value of this parameter determines the level of functionality required for publish/subscribe operations.

Valid values are 1 and 2.

The default value is 1.

-msgSelection
This parameter determines where message selection occurs.

Valid values are CLIENT and BROKER.

The default value is CLIENT.

-subStore
This parameter determines where WebSphere MQ messaging provider stores persistent data relating to active subscriptions.

Valid values are MIGRATE, QUEUE and BROKER.

The default value is MIGRATE.

-stateRefreshInt
The interval, in milliseconds, between refreshes of the long running transaction that detects when a subscriber loses its connection to the queue manager. This parameter is relevant only if **-subStore** parameter has the value QUEUE.

The value of this parameter must be a positive integer.

The default value is 60,000.

-cleanupLevel
The cleanup level for BROKER or MIGRATE subscription stores.

Valid values are SAFE, NONE, ASPROP, and STRONG.

The default value is SAFE.

-cleanupInterval
The interval between background executions of the publish/subscribe cleanup utility.

The value of this parameter must be a positive integer.

The default value is 3,600,000.

-wildcardFormat
This parameter determines which sets of characters are interpreted as topic wildcards.

Valid values are Topic or Char.

The default value is Char.

-sparseSubs
This parameter controls the message retrieval policy of a TopicSubscriber object.

The value of this parameter must be true or false

The default value is false.

-brokerQmgr
The name of the queue manager on which the broker is running.

-clonedSubs
This parameter determines whether two or more instances of the same durable topic subscriber can run simultaneously.

The value of this parameter must be ENABLED or DISABLED

The default value is DISABLED.

-qmgrSvrconnChannel
The SVRCONN channel to use when connecting to WebSphere MQ.

Use this parameter to create **explicitly defined** activation specifications.

The default value is SYSTEM.DEF.SVRCONN.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

-brokerCCDurSubQueue
The name of the queue from which a connection consumer receives durable subscription messages.

The default value is SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE.

-maxPoolSize
The maximum number of server sessions in the server session pool used by the connection consumer.

The value of this parameter must be a positive integer.

The default value is 10.

-messageSelector
A message selector expression specifying which messages are to be delivered.

The value of this parameter must be either the empty string or a valid SQL 92 statement.

-poolTimeout
The period of time, in milliseconds, that an unused server session is held open in the server session pool before being closed due to inactivity.

The value of this parameter must be a positive integer.

The default value is 300,000.

-startTimeout
The period of time, in milliseconds, within which delivery of a message to an MDB must start after the work to deliver the message has been scheduled. If this period of time elapses, the message is rolled back onto the queue.

The value of this parameter must be a positive integer.

The default value is 10,000.

-subscriptionDurability
This parameter determines whether a durable or nondurable subscription is used to deliver messages to an MDB that is subscribing to the topic.

The value of this parameter must be Durable or Nondurable

The default value is Nondurable.

-subscriptionName
The name of the durable subscription.

-customProperties
This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider activation specification implementation. Typically, custom properties are used to set attributes of the activation specification which are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:
- In Jython: [*name value*]
- In Jacl: {*name value*}

For example, **-WAS_EndpointInitialState** is a custom property that can be used with the **-customProperties** parameter. The value of **-WAS_EndpointInitialState** must be ACTIVE or INACTIVE. **-WAS_EndpointInitialState** determines whether the endpoint is activated when the endpoint is registered. If the parameter is set to active, message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.

This parameter should be ignored for subsequent activation or deactivation via the J2CMessageEndpoint MBean.

-localAddress
This parameter specifies either or both of the following:
- the local network interface
- the local port, or range of local ports

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

-sslType
This parameter determines the configuration, if any, to use when applying SSL encryption to the network connection to the queue manager.

The value of this parameter must be CENTRAL, SPECIFIC or NONE

The **-sslConfiguration** parameter is not valid unless this parameter is set to SPECIFIC.

The default value is NONE.

-sslConfiguration
The name of the SSL configuration to use when using SSL to secure network connections to the queue manager.

Do not specify this parameter unless the parameter **-sslType** is assigned the value SPECIFIC.

The value of this parameter must correspond to an SSL configuration.

There is no default value.

-stopEndpointIfDeliveryFails
This parameter indicates whether the endpoint should be stopped if message delivery fails the number of times specified by the **failureDeliveryCount** property.

The value of this parameter must be true or false.

The default value is true.

-failureDeliveryCount
This parameter specifies the number of sequential delivery failures that are allowed before the endpoint is suspended. This value is only used if **stopEndpointIfDeliveryFails** is true.

The value of this parameter must be a non-negative integer.

The default value is 0, which means that the endpoint is stopped the first time it fails.

Minimal activation specification definition

The following example creates an activation specification, specifying the minimum number of parameters. Due to the default values assumed for the unspecified parameters, MDBs deployed by using this activation specification are co-located with a generic queue manager installed on the same node.

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.createWMQActivationSpec("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name spec1 -jndiName jms/as/spec1
 -destinationJndiName jms/queues/q1 -destinationType javax.jms.Queue"])
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createWMQActivationSpec
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name spec1 -jndiName jms/as/spec1 -destinationJndiName jms/queues/q1
-destinationType javax.jms.Queue}
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

Explicit activation specification definition

The following example creates an activation specification for which the user must specify and maintain all the parameters used for establishing a connection to WebSphere MQ.

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.createWMQActivationSpec("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name spec2 -jndiName 'jms/as/spec2'
 -destinationJndiName 'jms/topics/t2' -destinationType javax.jms.Topic
 -description 'Must remember to keep each of these activation specifications in
 sync with the WebSphere MQ queue manager to which they refer' -qmgrName QM1
 -qmgrHostname 192.168.0.22 -qmgrPort 1415 -qmgrSvrconnChannel QM1.SVRCONN"])
spec2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234987)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createWMQActivationSpec
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name spec2 -jndiName "jms/as/spec2" -destinationJndiName "jms/topics/t2"
-destinationType javax.jms.Topic -description "Must remember to keep each
of these activation specifications in sync with the WebSphere MQ queue manager
to which they refer" -qmgrName QM1 -qmgrHostname 192.168.0.22 -qmgrPort 1415
-qmgrSvrconnChannel QM1.SVRCONN}
spec2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234987)
```

Activation specification definition specifying a CCDT

The following example creates an activation specification that uses a CCDT to locate the queue manager to connect to.

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode019994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)")

wsadmin>AdminTask.createWMQActivationSpec("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name spec3 -jndiName 'jms/as/spec3'
 -destinationJndiName 'jms/queue/q3' -destinationType javax.jms.Queue
 -ccdtUrl 'http://gorillaaction:9080/ccdt/amqclchl.tab' -ccdtQmgrName QM3"])
spec3(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234988)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createWMQActivationSpec
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name spec3 -jndiName "jms/as/spec3" -destinationJndiName "jms/queue/q3"
-destinationType javax.jms.Queue -ccdtUrl "http://gorillaaction:9080/ccdt/
amqclchl.tab" -ccdtQmgrName QM3}
spec3(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234988)
```

Creating an activation specification with the -WAS_EndpointInitialState custom property set to ACTIVE

- The following example creates an activation specification with the WAS_EndpointInitialState custom property activated, using Jython:

```
wsadmin>attrs = '[[name "WAS_EndpointInitialState"] [required "false"] [type "java.lang.String"] [value "ACTIVE"]]
wsadmin>AdminConfig.getid("/Node:myNode01")
myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)'
```

```
wsadmin>theActSpec = AdminTask.createWMQActivationSpec("myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)",
'-name testas -jndiName testas -destinationJndiName testq -destinationType javax.jms.Queue
-customProperties [[WAS_EndpointInitialState Active]]')
```

deleteWMQActivationSpec command

Use this command to delete a WebSphere MQ messaging provider activation specification at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the deleteWMQActivationSpec command to delete a WebSphere MQ messaging provider activation specification defined at the scope at which the command is issued.

Target object

A WebSphere MQ messaging provider activation specification at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQActivationSpecs("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
unwantedSpec(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

```
wsadmin>AdminTask.deleteWMQActivationSpec("unwantedSpec(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#J2CActivationSpec_1098737234986)")
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
unwantedSpec(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)

wsadmin>$AdminTask deleteWMQActivationSpec
unwantedSpec(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

listWMQActivationSpecs command

Use the listWMQActivationSpecs command to list WebSphere MQ messaging provider activation specifications.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the listWMQActivationSpecs command to list all of the WebSphere MQ messaging provider activation specifications defined at the scope at which the command is issued.

Target object

WebSphere MQ messaging provider activation specifications at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQActivationSpecs("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

modifyWMQActivationSpec command

Use the modifyWMQActivationSpec command to change certain parameters of a WebSphere MQ messaging provider activation specification.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the modifyWMQActivationSpec command to modify a WebSphere MQ messaging provider activation specification defined at the scope at which the command is issued.

Note: You cannot change the type of an activation specification. For example, you cannot create an activation specification where you enter all the configuration information manually and then modify it to use a CCDT.

For a CCDT-based activation specification, you cannot modify of the following parameters:

- **qmgrName**
- **qmgrHostname**
- **qmgrPortNumber**
- **qmgrSrvconnChannel**

- **transportChain**
- **wmqTransportType**

For a generic activation specification, you cannot modify any of the following parameters:

- **ccdtUrl**
- **ccdtQmgrName**

Target object

A WebSphere MQ messaging provider activation specification at the specific scope.

Required parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider activation specification.

Optional parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider activation specification.

Note: If either the **-qmgrHostname** or **-qmgrPortNumber** parameters are specified without the **-connectionNameList** parameter being specified, then it is assumed that a connection name list should no longer be used to connect to WebSphere MQ and that the specified host name and port number information should be used instead. As a result of this the **-connectionNameList** parameter is set to blank.

Note the behavior of this command on the **-customProperties** parameter.

-customProperties

This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider activation specification implementation. Typically, custom properties are used to set attributes of the activation specification which are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:

- In Jython: [*name value*]
- In Jacl: {*name value*}

For example, **-WAS_EndpointInitialState** is a custom property that can be used with the **-customProperties** parameter. The value of **-WAS_EndpointInitialState** must be ACTIVE or INACTIVE. **-WAS_EndpointInitialState** determines whether the endpoint is activated when the endpoint is registered. If the parameter is set to active, message consumption begins from the JMS destination as soon as the activation specification is used for a message-driven bean to connect with the destination.

This parameter should be ignored for subsequent activation or deactivation via the J2CMessageEndpoint MBean.

New name/value pairs are added to the existing set of custom properties using the following rules:

- If the existing set of properties does not contain a property with the same name as that supplied as part of a modify command, the supplied property is added to the set of custom properties, unless the custom property has no value specified, when it is disregarded.

- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, and the modify command also specifies a value for the property, the existing value is replaced by the supplied value.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, but the modify command does not specify a value for the property, the property with the same name is deleted from the existing set of custom properties.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQActivationSpecs("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)


wsadmin>AdminTask.modifyWMQActivationSpec("spec1(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#J2CActivationSpec_1098737234986)",
["-destinationJndiName jms/topics/t5 -destinationType javax.jms.Topic"])
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)


wsadmin>$AdminTask modifyWMQActivationSpec
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
{-destinationJndiName jms/topics/t5 -destinationType javax.jms.Topic}
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
```

- The following example modifies an activation specification by activating the WAS_EndpointInitialState custom property, using Jython:

```
wsadmin>AdminConfig.getid("/Node:myNode01")
'myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)'
wsadmin>wsadmin>AdminTask.listWMQActivationSpecs("myNode01(cells/myCell01/nodes/myNode01|node.xml#Node_1)")
'newas(cells/myCell01/nodes/myNode01|resources.xml#J2CActivationSpec_1298546034140)'
wsadmin>AdminTask.modifyWMQActivationSpec("newas(cells/myCell01/nodes/myNode01|resources.xml
#J2CActivationSpec_1298546034140)", '-customProperties [[WAS_EndpointInitialState ACTIVE]]')
```

showWMQActivationSpec command

Use the showWMQActivationSpec command to display information about a specific WebSphere MQ messaging provider activation specification.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the showWMQActivationSpec command to display all the parameters, and their values, associated with a particular WebSphere MQ messaging provider activation specification.

Target object

A WebSphere MQ messaging provider activation specification at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQActivationSpecs("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)

wsadmin>AdminTask.showWMQActivationSpec("spec1(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#J2CActivationSpec_1098737234986)")
{cleanupLevel=SAFE, useConnectionPooling=true, port=1414, maxPoolDepth=10,
channel=channel1, transportType=CLIENT, subscriptionStore=MIGRATE,
messageSelection=CLIENT, cleanupInterval=3600000,
brokerCCSubQueue=SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE, name=spec1, CCSID=819,
useJNDI=true, hostName=localhost, rescanInterval=5000, headerCompression=NONE,
brokerCCDurSubQueue=SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE, queueManager=QMGR1,
messageCompression=NONE, startTimeout=10000,  destinationJndiName=jms/q1,
poolTimeout=300000, sslType=NONE,
destinationType=javax.jms.Queue, brokerSubQueue=SYSTEM.JMS.ND.SUBSCRIBER.QUEUE,
sslResetCount=0, brokerControlQueue=SYSTEM.BROKER.CONTROL.QUEUE,
stateRefreshInt=60000, cloneSupport=DISABLED, jndiName=jms/as1,
sparseSubscriptions=false, authenticationAlias=null, failIfQuiesce=true,
description=, brokerVersion=1}
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
```

```
wsadmin>$AdminTask listWMQActivationSpecs
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)

wsadmin>$AdminTask showWMQActivationSpec
spec1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
J2CActivationSpec_1098737234986)
{cleanupLevel=SAFE, useConnectionPooling=true, port=1414, maxPoolDepth=10,
channel=channel1, transportType=CLIENT, subscriptionStore=MIGRATE,
messageSelection=CLIENT, cleanupInterval=3600000,
brokerCCSubQueue=SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE, name=spec1, CCSID=819,
useJNDI=true, hostName=localhost, rescanInterval=5000, headerCompression=NONE,
brokerCCDurSubQueue=SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE, queueManager=QMGR1,
messageCompression=NONE, startTimeout=10000,  destinationJndiName=jms/q1,
poolTimeout=300000, sslType=NONE,
destinationType=javax.jms.Queue, brokerSubQueue=SYSTEM.JMS.ND.SUBSCRIBER.QUEUE,
sslResetCount=0, brokerControlQueue=SYSTEM.BROKER.CONTROL.QUEUE,
stateRefreshInt=60000, cloneSupport=DISABLED, jndiName=jms/as1,
sparseSubscriptions=false, authenticationAlias=null, failIfQuiesce=true,
description=, brokerVersion=1}
```

createWMQConnectionFactory command

Use the createWMQConnectionFactory command to create a connection factory for the WebSphere MQ messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the createWMQConnectionFactory command to create a WebSphere MQ messaging provider connection factory at a specific scope.

You cannot create a WebSphere MQ messaging provider connection factory under either of the following conditions:

- A WebSphere MQ messaging provider connection factory already exists with the same name, at the same scope.
- The JNDI name clashes with another entry in WebSphere Application Server JNDI.

Target object

The scope of the WebSphere MQ messaging provider at which the WebSphere MQ messaging provider connection factory is to be created.

Required parameters

-name
The administrative name assigned to this WebSphere MQ messaging provider connection factory.

-jndiName
The name and location used to bind this object into WebSphere Application Server JNDI.

-type
Use this parameter to determine whether a unified connection factory, a queue connection factory or a topic connection factory is to be created.

Enter one of the following values:

- CF
- QCF
- TCF

CF is the default value.

If you specify QCF, you cannot specify any of the following parameters:

- **-brokerCtrlQueue**
- **-brokerSubQueue**
- **-brokerCCSubQueue**
- **-brokerVersion**
- **-brokerPubQueue**
- **-tempTopicPrefix**
- **-pubAckWindow**
- **-subStore**
- **-stateRefreshInt**
- **-cleanupLevel**
- **-sparesSubs**
- **-wildcardFormat**
- **-brokerQmgr**
- **-clonedSubs**
- **-msgSelection**

If you specify TCF, you cannot specify any of the following parameters:

- **-msgRetention**
- **-rescanInterval**
- **-tempQueuePrefix**
- **-modelQueue**
- **-replyWithRFH2**

Optional parameters

-description
An administrative description assigned to the connection factory.

-ccdtUrl
A URL to a client channel definition table to use, for this connection factory, when contacting WebSphere MQ.

Use this parameter to create a ccdtURL connection factory

Do not specify this parameter in conjunction with the following parameters: **-qmgrName**, **-qmgrHostname**, **-qmgrPortNumber**, **-qmgrSvrconnChannel**, **–wmqTransportType** or **-localAddress**.

-ccdtQmgrName
A queue manager name, used to select one or more entries from a client channel definition table.

Do not specify this parameter in conjunction with the following parameters: **-qmgrName**, **-qmgrHostname**, **-qmgrPortNumber**, **-qmgrSvrconnChannel**, **–wmqTransportType**, or **-localAddress**.

-qmgrName
The name of the queue manager to use, for this connection factory, when contacting WebSphere MQ.

Use this parameter to create a generic connection factory.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

-wmqTransportType
This parameter determines the way in which a connection is established to WebSphere MQ for this connection factory.

Use this parameter to create a generic connection factory.

Enter one of the following values:

- BINDINGS
- BINDINGS_THEN_CLIENT
- CLIENT

The default value is BINDINGS_THEN_CLIENT.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

For more information about configuring a transport type of BINDINGS_THEN_CLIENT or BINDINGS, refer to “Configuring the WebSphere MQ messaging provider with native libraries information” on page 723.

-qmgrHostname
The host name to use, for this connection factory, when attempting a client mode connection to WebSphere MQ. It must be a valid TCP/IP host name or IPv4 or IPv6 address.

The default value is the local host.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

For information on setting the **-qmgrHostname** parameter in conjunction with the **-connectionNameList** parameter, see the description of the **-connectionNameList** parameter.

-qmgrPortNumber
The port number to use, for this connection factory, when attempting a client mode connection to WebSphere MQ.

Enter an integer value in the range 1 - 65536 (inclusive).

The default value is 1414.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

For information on setting the **-qmgrPortNumber** parameter in conjunction with the **-connectionNameList** parameter, see the description of the **-connectionNameList** parameter.

-connectionNameList

The connection name list specifying the host name and port details to use when you want the connection factory to connect to a multi-instance queue manager.

Note: You must only use the **-connectionNameList** parameter to allow a connection to a multi-instance queue manager. Using the **-connectionNameList** parameter to connect to a non-multi-instance queue manager can jeopardize transaction integrity.

The **-connectionNameList** parameter must be entered as a comma separated list of host names and ports in the following format:

host(*port*),*host*(*port*)

For *host* enter a valid TCP/IP host name, IPv4 or IPv6 address.

For *port* enter an integer value between 1 and 65536 (inclusive). Specifying a value for *port* is optional. When you do not specify a value, *port* defaults to 1414.

For example: `localhost(1234),remotehost1(1234),remotehost2`

When you specify the **-connectionNameList** parameter, the **-qmgrHostname** and **-qmgrPortNumber** parameters are automatically set to the host name and port number of the first entry in the connection name list. This overrides any values that you previously specified in the **-qmgrHostname** and **-qmgrPortNumber** parameters.

In the preceding example this would mean that **-qmgrHostname** would be `localhost` and **-qmgrPortNumber** would be `1234`.

The **-connectionNameList** parameter is only valid for use in WebSphere Application Server Version 8.0. Attempting to specify the **-connectionNameList** parameter on a WebSphere MQ messaging provider connection factory which is defined at a server or node scope that is running on a version of WebSphere Application Server earlier than Version 8.0 results in an error message when you run the following commands:

- createWMQConnectionFactory
- modifyWMQConnectionFactory

If a WebSphere MQ messaging provider connection factory that is based on a connection name list is used by an application client or server that is running a version of WebSphere Application Server earlier than Version 8.0, the **-connectionNameList** information is not used. Instead, the values specified in the **-qmgrHostname** and **–qmgrPortNumber** are used, and are set to the relevant values from the first entry in the **connectionNameList** parameter.

You must not specify the **connectionNameList** parameter in conjunction with the **-ccdtUrl** or **-ccdtQmgrName** parameters.

Note: If you use the**-connectionNameList** parameter with a centrally managed SSL configuration, the host name and port number information used to select the appropriate SSL configuration is based on the first entry in the connection name list, regardless of which entry in the list is actually used to connect to the queue manager. This is because each instance of a multi-instance queue manager should be using the same SSL configuration, for a given server connection channel, regardless of which instance is actually running.

For more information on using multi-instance queue managers see the WebSphere MQ information center.

-containerAuthAlias
The container-managed authentication alias, defined to the cell, from which security credentials are used to establish a connection to WebSphere MQ.

-componentAuthAlias
The component-managed authentication alias, defined to the cell, from which security credentials are used to establish a connection to WebSphere MQ.

-clientId
The client identifier used for connections started by using this connection factory.

-providerVersion
This parameter determines the minimum version, and capabilities of the queue manager.

Enter values in one of the following formats:

- *n*
- *n.n*
- *n.n.n*
- *n.n.n.n*

where *n* is an integer greater than or equal to zero.

For example 6.0.0.0

-sslCrl
This parameter specifies a list of LDAP servers that are used to provide certificate revocation information if this connection factory establishes an SSL based connection to WebSphere MQ.

-sslResetCount
This parameter is used when the connection factory establishes an SSL connection to the queue manager. This parameter determines how many bytes to transfer before resetting the symmetric encryption key that is used for the SSL session.

Enter a value in the range 0 through 999,999,999.

The default value is 0.

-sslPeerName
This parameter is used when the connection factory establishes an SSL connection to the queue manager. The value is compared with the distinguished name present in the peer's certificate.

-rcvExit
A comma-separated list of receive exit class names.

-rcvExitInitData
Initialization data to pass to the receive exit.

Do not specify this parameter unless you specify the **-rcvExit** parameter.

-sendExit
A comma-separated list of send exit class names.

-sendExitInitData
Initialization data to pass to the send exit.

Do not specify this parameter unless you specify the **-sendExit** parameter.

-secExit
A security exit class name.

-secExitInitData
Initialization data to pass to the security exit.

Do not specify this parameter unless you specify the **-secExit** parameter.

-compressHeaders
This parameter determines if message headers are compressed.

Enter one of the following values:

- NONE
- SYSTEM

The default value is NONE.

-compressPayload
This parameter determines if message payloads are compressed.

Enter one of the following values:

- NONE
- RLE
- ZLIBFAST
- ZLIBHIGH

The default value is NONE.

-msgRetention
This parameter determines if the connection consumer keeps unwanted messages on the input queue.

Enter one of the following values:

- YES
- NO

where YES specifies that the connection consumer keeps unwanted messages on the input queue, and NO specifies that the messages are disposed of according to their disposition options.

The default value is YES.

-pollingInterval
This property is applicable in the client container only.

If each message listener within a session has no suitable message on its queue, this parameter is the maximum interval, in milliseconds, that elapses before each message listener tries again to get a message from its queue. If it frequently happens that no suitable message is available for any of the message listeners in a session, consider increasing the value of this parameter.
The default value is 5000.

-rescanInterval
When a message consumer in the point-to-point domain uses a message selector to select which messages it is to receive, the JMS client searches the WebSphere MQ queue for suitable messages in the sequence determined by the MsgDeliverySequence attribute of the queue. When the client finds a suitable message and delivers it to the consumer, the client resumes the search for the next suitable message from its current position in the queue. The client continues to search the queue in this way until it reaches the end of the queue, or until the interval of time in milliseconds, as determined by the value of this **-rescanInterval** parameter has expired. In each case, the client returns to the beginning of the queue to continue its search, and a new time interval commences

This parameter must be a positive integer value.

The default value is 5000.

-ccsid
The coded character set identifier (CCSID) to be used on connections.

The value of this parameter must be a positive integer. See the “WebSphere MQ messaging provider connection factory advanced properties” on page 772 for more details.

The default value is 819.

-failIfQuiescing
This parameter determines the behavior of certain calls to the queue manager when the queue manager is put into quiescing state.

The value of this parameter must be true or false.

true specifies that calls to certain methods fail if the queue manager is in a quiescing state. If an application detects that the queue manager is quiescing, the application can complete its immediate task and close the connection, allowing the queue manager to stop.

false specifies that no methods fail if the queue manager is in a quiescing state. If you specify this value, an application cannot detect that the queue manager is quiescing. The application might continue to perform operations against the queue manager, and therefore prevent the queue manager from stopping.

The default value is true.

-brokerCtrlQueue
The name of the broker control queue to use if this connection factory is to subscribe to a topic.

The default value is SYSTEM.BROKER.CONTROL.QUEUE.

-brokerSubQueue
The name of the queue to use for obtaining subscription messages if this connection factory is to subscribe to a topic.

The default value is SYSTEM.JMS.ND.SUBSCRIBER.QUEUE.

-brokerCCSubQueue
The name of the queue from which non-durable subscription messages are retrieved for a ConnectionConsumer.

The default value is SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE.

-brokerVersion
The value of this parameter determines the level of functionality required for publish/subscribe operations.

Valid values are 1 and 2.

The default value is 1.

-msgSelection
This parameter determines where message selection occurs.

Valid values are CLIENT and BROKER.

The default value is CLIENT.

-subStore
This parameter determines where WebSphere MQ messaging provider stores persistent data relating to active subscriptions.

Valid values are MIGRATE, QUEUE and BROKER.

The default value is MIGRATE.

-stateRefreshInt
The interval, in milliseconds, between refreshes of the long running transaction that detects when a subscriber loses its connection to the queue manager. This parameter is relevant only if **-subStore** parameter has the value QUEUE.

The value of this parameter must be a positive integer.

The default value is 60,000.

-cleanupLevel
The cleanup level for BROKER or MIGRATE subscription stores

Valid values are SAFE, NONE, ASPROP, and STRONG.

The default value is SAFE.

-cleanupInterval
The interval between background executions of the publish/subscribe cleanup utility.

The value of this parameter must be a positive integer.

The default value is 3,600,000.

-wildcardFormat
This parameter determines which sets of characters are interpreted as topic wildcards.

Valid values are Topic or Char.

The default value is Topic.

-sparseSubs
This parameter controls the message retrieval policy of a TopicSubscriber object.

The value of this parameter must be true or false

The default value is false.

-brokerQmgr
The name of the queue manager that is running the broker, if it is not the same as the queue manager to which the connection factory connects.

There is no default value.

-clonedSubs
This parameter determines whether two or more instances of the same durable topic subscriber can run simultaneously

The value of this parameter must be ENABLED or DISABLED

The default value is DISABLED.

-customProperties
This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider connection factory implementation. Typically, custom properties are used to set attributes of the connection factory that are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:
- In Jython: [*name value*]
- In Jacl: {*name value*}

-qmgrSvrconnChannel
The SVRCONN channel to use when connecting to WebSphere MQ.

Use this parameter to create an **explicitly defined** connection factory.

The default value is SYSTEM.DEF.SVRCONN.

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

-support2PCProtocol
This parameter determines if the connection factory acts as a resource that is capable of participation in distributed two-phase commit processing.

The value of this parameter must be True or False.

The default value True specifies that the connection factory acts as a resource that is capable of participation in distributed two-phase commit processing.

-modelQueue
The name of the WebSphere MQ model queue whose definition is used as a basis when creating JMS temporary destinations.

The default value is SYSTEM.DEFAULT.MODEL.QUEUE.

-tempQueuePrefix
The prefix to apply to WebSphere MQ temporary queues that are used to represent JMS temporary queue type destinations.

There is no default value.

-tempTopicPrefix
The prefix to apply to the names generated for temporary topics. This parameter is only valid for connection factories or topic connection factories.

There is no default value.

-replyWithRFH2
This parameter determines whether, when sending a reply message to the reply-to queue obtained from a message that does not include an RFH version 2 header, a RFH version 2 header is included in the reply message.

The value of this parameter must be ALWAYS or AS_REPLY_DEST

The default value is AS_REPLY_DEST.

-brokerPubQueue
The name of the queue to which to send publication messages when using queue based brokering.

The default value is SYSTEM.BROKER.DEFAULT.STREAM.

-pubAckInterval
The number of publications to send to a queue based broker before sending a publication that solicits an acknowledgement.

The value of this parameter must be a positive integer greater than zero.

The default value is 25.

-sslType
This parameter determines the configuration, if any, to use when applying SSL encryption to the network connection to the queue manager.

The value of this parameter must be CENTRAL, SPECIFIC or NONE

The default value is NONE.

The **sslConfiguration** parameter is not valid unless this parameter is set to the value SPECIFIC.

-sslConfiguration
The name of the SSL configuration to use when using SSL to secure network connections to the queue manager.

The value of this parameter must correspond to an SSL configuration.

Do not specify this parameter unless the parameter **-sslType** is assigned the value SPECIFIC.

-localAddress
This parameter specifies either or both of the following:

- the local network interface
- the local port, or range of local ports

Do not specify this parameter in conjunction with the following parameters: **-ccdtUrl** or **-ccdtQmgrName**.

-mappingAlias
The JAAS mapping alias used when determining which security credentials to use when establishing a connection to WebSphere MQ.

The default value is DefaultPrincipleMapping.

-xaRecoveryAuthAlias
The authentication alias from which credentials are taken and used to connect to WebSphere MQ for XA recovery.

There is no default value.

Minimal connection factory definition

The following example creates an connection factory, specifying the minimum number of parameters. Due to the default values assumed for the unspecified parameters, applications using this connection factory expect to be co-located with a queue manager installed on the same node.

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createWMQConnectionFactory
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name cf1 -jndiName "jms/cf/cf1" -type CF}
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1205322636000)
```

Explicitly defined connection factory

The following example creates an connection factory for which the user must specify and maintain all the parameters used for establishing a connection to WebSphere MQ.

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.createWMQConnectionFactory("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)", ["-name cf2
 -jndiName 'jms/cf/cf2' -type CF -description 'Must remember to keep each
of these connection factories in sync with the WebSphere MQ queue manager
to which they refer' -qmgrName QM1 -qmgrHostname 192.168.0.22 -qmgrPort 1415
 -qmgrSvrconnChannel QM1.SVRCONN"])
cf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_120532263601)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createWMQConnectionFactory
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name cf2 -jndiName "jms/cf/cf2" -type CF -description "Must remember to
keep each of these connection factories in sync with the WebSphere MQ queue
manager to which they refer" -qmgrName QM1 -qmgrHostname 192.168.0.22
-qmgrPort 1415 -qmgrSvrconnChannel QM1.SVRCONN}
cf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_120532263601)
```

Connection factory definition specifying a CCDT

The following example creates an connection factory that uses a CCDT to locate the queue manager to connect to.

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.createWMQConnectionFactory("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)", ["-name cf3 -jndiName
 'jms/cf/cf3' -type CF -ccdtUrl 'http://gorillaaction:9080/ccdt/amqclchl.tab'
 -ccdtQmgrName QM3"])
cf3(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_120532263606)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createWMQConnectionFactory
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name cf3 -jndiName "jms/cf/cf3" -type CF -ccdtUrl
"http://gorillaaction:9080/ccdt/amqclchl.tab" -ccdtQmgrName QM3}
cf3(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_120532263606)
```

deleteWMQConnectionFactory command

Use the deleteWMQConnectionFactory command to delete a WebSphere MQ messaging provider connection factory at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the deleteWMQConnectionFactory command to delete a WebSphere MQ messaging provider connection factory defined at the scope at which the command is issued.

Target object

A WebSphere MQ messaging provider connection factory at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQConnectionFactories("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)")
unwantedCF(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)

wsadmin>AdminTask.deleteWMQConnectionFactory("unwantedCF(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#MQConnectionFactory_1098737234986)")
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQConnectionFactories
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
unwantedCF(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)

wsadmin>$AdminTask deleteWMQConnectionFactory
unwantedCF(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)
```

listWMQConnectionFactories command

Use the listWMQConnectionFactories command to list WebSphere MQ messaging provider connection factories.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the listWMQConnectionFactories command to list all of the WebSphere MQ messaging provider connection factories defined at the scope at which the command is issued.

Target object

WebSphere MQ messaging provider connection factories at the specific scope.

Required parameters

None.

Optional parameters

-type
Use this parameter to determine which type of connection factory is listed. If it is omitted all connection factories are shown at the appropriate scope.

Valid values are:

- CF to list only common connection factories
- QCF to list only queue connection factories
- TCF to list only topic connection factories

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQConnectionFactories("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)")
cf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQConnectionFactories
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
cf2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)
```

modifyWMQConnectionFactory command

Use the modifyWMQConnectionFactory command to change certain parameters of a WebSphere MQ messaging provider connection factory.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the modifyWMQConnectionFactory command to modify a WebSphere MQ messaging provider connection factory defined at the scope at which the command is issued.

Note: When modifying a WebSphere MQ messaging provider connection factory, there is an interaction between the **mappingAlias** and **containerAuthAlias** parameters. This interaction occurs if the **containerAuthAlias** parameter is specified but the **mappingAlias** is not specified. In this situation, the **mappingAlias** parameter is automatically set to the value DefaultPrincipleMapping.

Target object

A WebSphere MQ messaging provider connection factory at the specific scope.

Required parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider connection factory.

Optional parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider connection factory.

Note: If either the **-qmgrHostname** or **-qmgrPortNumber** parameters are specified without the **-connectionNameList** parameter being specified, then it is assumed that a connection name list should no longer be used to connect to WebSphere MQ and that the specified host name and port number information should be used instead. As a result of this the **-connectionNameList** parameter is set to blank.

Note the behavior of this command on the **-customProperties** parameter.

-customProperties
This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider connection factory implementation. Typically, custom properties are used to set attributes of the connection factory that are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:
- In Jython: [*name value*]
- In Jacl: {*name value*}

New name/value pairs are added to the existing set of custom properties using the following rules:

- If the existing set of properties does not contain a property with the same name as that supplied as part of a modify command, the supplied property is added to the set of custom properties, unless the custom property has no value specified, when it is disregarded.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, and the modify command also specifies a value for the property, the existing value is replaced by the supplied value.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command but the modify command does not specify a value for the property, the property with the same name is deleted from the existing set of custom properties.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQConnectionFactories("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)")
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)

wsadmin>AdminTask.modifyWMQConnectionFactory("cf1(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#MQConnectionFactory_1098737234986)", ["-description
 'My new description'"])
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQConnectionFactories
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)

wsadmin>$AdminTask modifyWMQConnectionFactory
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986) {-description "My new description"}
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)
```

showWMQConnectionFactory command

Use the showWMQConnectionFactory command to display information about a specific WebSphere MQ messaging provider connection factory.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the showWMQConnectionFactory command to display all the parameters, and their values, associated with a particular WebSphere MQ messaging provider connection factory.

Target object

A WebSphere MQ messaging provider connection factory at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQConnectionFactories("9994GKCNode01(cells/
9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)")
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)

wsadmin>AdminTask.showWMQConnectionFactory("cf1(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#MQConnectionFactory_1098737234986)")
{ name=cf1 jndiName=jms/cf/cf1 description= qmgrName=QMGR1
qmgrSvrconnChannel=TO.QMGR1 qmgrHostname=localhost qmgrPortNumber=1414
wmqTransportType=bindingsThenClient authAlias= clientId="Bob's Magic Client"
providerVersion= sslCrl= sslResetCount= sslPeerName= rcvExit=
rcvExitInitData= sendExit= sendExitInitData= secExit= secExitInitData=
compressHeaders=NONE compressPayload=NONE msgRetention=true
pollingInterval=5000 rescanInterval=5000 maxBatchSize=10 ccsid=819
failIfQuiescing=true brokerCtrlQueue=
brokerSubQueue=SYSTEM.BROKER.CONTROL.QUEUE
brokerCCSubQueue=SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
brokerVersion=1 msgSelection=CLIENT subStore=MIGRATE stateRefreshInt=60000
cleanupInterval=360000000 cleanupLevel=SAFE wildcardFormat=CHAR
sparseSubs=false brokerQmgr= clonedSubs=true}
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQConnectionFactories
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQConnectionFactory_1098737234986)

wsadmin>$AdminTask showWMQConnectionFactory
cf1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
aclMQConnectionFactory_1098737234986)
{ name=cf1 jndiName=jms/cf/cf1 description= qmgrName=QMGR1
qmgrSvrconnChannel=TO.QMGR1 qmgrHostname=localhost qmgrPortNumber=1414
```

```
wmqTransportType=bindingsThenClient authAlias= clientId="Bob's Magic Client"
providerVersion= sslCrl= sslResetCount= sslPeerName= rcvExit=
rcvExitInitData= sendExit= sendExitInitData= secExit= secExitInitData=
compressHeaders=NONE compressPayload=NONE msgRetention=true
pollingInterval=5000 rescanInterval=5000 maxBatchSize=10 ccsid=819
failIfQuiescing=true brokerCtrlQueue=
brokerSubQueue=SYSTEM.BROKER.CONTROL.QUEUE
brokerCCSubQueue=SYSTEM.JMS.ND.CC.SUBSCRIBER.QUEUE
brokerVersion=1 msgSelection=CLIENT subStore=MIGRATE stateRefreshInt=60000
cleanupInterval=360000000 cleanupLevel=SAFE wildcardFormat=CHAR
sparseSubs=false brokerQmgr= clonedSubs=true}
```

createWMQTopic command

Use the createWMQTopic command to create a JMS topic destination for the WebSphere MQ messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the createWMQTopic command to create a WebSphere MQ messaging provider topic type destination at a specific scope.

You cannot create a WebSphere MQ messaging provider topic type destination under either of the following conditions:

- A WebSphere MQ messaging provider topic type destination already exists with the same name, at the same scope.
- The JNDI name clashes with another entry in WebSphere Application Server JNDI.

Target object

The scope of the WebSphere MQ messaging provider at which the WebSphere MQ messaging provider topic type destination is to be created.

Required parameters

-name
The administrative name assigned to this WebSphere MQ messaging provider topic type destination.

-jndiName
The name used to bind this object into WebSphere Application Server JNDI.

-topicName
The name of the WebSphere MQ topic where publications are received from, or sent to, when this destination definition is used.

Optional parameters

-description
An administrative description assigned to the topic type destination.

-persistence
This parameter determines the level of persistence used to store messages sent to this destination.

Enter one of the following case-sensitive values:

- APP
- TDEF
- PERS
- NON
- HIGHT

APP is the default value.

-priority
The priority level to assign to messages sent to this destination.

Enter one of the following case-sensitive values:

- APP
- TDEF

or enter a positive integer in the range 0 to 9 (inclusive).

The default value is APP .

-expiry
The length of time after which messages that are sent to this destination expire and are dealt with according to their disposition options.

Enter one of the following case-sensitive values:

- APP
- UNLIM

or enter any positive integer.

The default value is APP.

-ccsid
The coded character set identifier (CCSID).

The value of this parameter must be a positive integer or blank. See the “WebSphere MQ messaging provider queue and topic advanced properties settings” on page 844 for more details.

The default value is 1208.

Leaving this field empty indicates that the default value must be used.

-useNativeEncoding
This parameter specifies whether to use native encoding or not. It can take a value true or false.

If it is set to true, the values of the **-integerEncoding**, **-decimalEncoding** and **-floatingPointEncoding** attributes are ignored.

If it is set to false, the encoding is specified by the **-integerEncoding**, **-decimalEncoding** and **-floatingPointEncoding** attributes.

-integerEncoding
The integer encoding setting for this queue.

Enter one of the following case-sensitive values: Normal, Reversed.

Normal is the default value.

-decimalEncoding
The decimal encoding setting for this queue.

Enter one of the following case-sensitive values: Normal, Reversed.

The default value is Normal.

-floatingPointEncoding
The floating point encoding setting for this queue.

Enter one of the following case-sensitive values: IEEENormal, IEEEReversed, z/OS

The default value is IEEENormal.

-useRFH2
This parameter determines whether an RFH version 2 header is appended to messages sent to this destination.

Enter one of the following case-sensitive values: true or false.

The default value is true.

-sendAsync
This parameter determines whether messages can be sent to this destination without queue manager acknowledging that they have arrived.

Enter one of the following case-sensitive values: YES, NO or TDEF.

The default value is YES.

-readAhead
This parameter determines whether messages for non-persistent consumers can be read ahead and cached.

Enter one of the following case-sensitive values: YES, NO or TDEF.

The default value is YES.

-readAheadClose
This property determines the behavior that occurs when closing a message consumer that is receiving messages asynchronously by using a message listener from a destination that has the **readAhead** parameter set to True.

When a value of deliverAll is specified, all read-ahead messages are delivered before closing the consumer.

When a value of deliverCurrent is specified, only in-progress messages are delivered before closing the consumer.

The default value is deliverCurrent.

-wildcardFormat
This parameter determines which sets of characters are interpreted as topic wildcards.

Valid values are Topic or Char.

The default value is Topic.

-brokerDurSubQueue
The name of the queue, defined to the queue manager, from which a connection consumer receives non-durable subscription messages.

The value of this parameter must be a valid queue name or left blank

The default value is SYSTEM.JMS.D.SUBSCRIBER.QUEUE.

-brokerCCDurSubQueue
The name of the queue, defined to the queue manager, from which a connection consumer receives durable subscription messages.

The value of this parameter must be a valid queue name or left blank

The default value is SYSTEM.JMS.D.CC.SUBSCRIBER.QUEUE.

-brokerPubQueue
The name of the queue, defined to the queue manager, to which publication messages are sent.

The value of this parameter must be a valid queue name or left blank

The default value is SYSTEM.BROKER.DEFAULT.STREAM.

-brokerPubQmgr
The name of the queue manager on which the broker is running.

The value of this parameter must be a valid queue manager name or left blank

There is no default value.

-brokerVersion
This parameter determines the level of functionality required for publish/subscribe operations.

The value of this parameter must be V1 or V2.

The default value is V1.

-customProperties
This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider topic type destination implementation. Typically, custom properties are used to set attributes of the topic type destination that are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:
- In Jython: [*name value*]
- In Jacl: {*name value*}

The following example creates a topic definition by specifying the minimum number of parameters.

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
 9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.createWMQTopic("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name T1 -jndiName jms/topic/t1
 -topicName woodland/creatures/badger"])
T1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
 9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask createWMQTopic
```

```
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name T1 -jndiName jms/topic/t1 -topicName woodland/creatures/badger}
T1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
```

deleteWMQTopic command

Use this command to delete a WebSphere MQ messaging provider topic at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the deleteWMQTopic command to delete a WebSphere MQ messaging provider topic defined at the scope at which the command is issued.

Target object

A WebSphere MQ messaging provider topic at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQTopics("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
unwantedTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
```

```
wsadmin>$AdminTask deleteWMQTopic
unwantedTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQTopics
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
unwantedTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)

wsadmin>$AdminTask deleteWMQTopic
unwantedTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
```

listWMQTopics command

Use the listWMQTopics command to list WebSphere MQ messaging provider topics.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the listWMQTopics command to list all of the WebSphere MQ messaging provider topics defined at the scope at which the command is issued.

Target object

WebSphere MQ messaging provider topics at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQTopics("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
aaaTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
bbbTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234987)
cccTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234988)
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQTopics
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
aaaTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
bbbTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234987)
cccTopic(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234988)
```

modifyWMQTopic command

Use the modifyWMQTopic command to change certain parameters of a WebSphere MQ messaging provider topic.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the modifyWMQTopic command to modify a WebSphere MQ messaging provider topic defined at the scope at which the command is issued.

Note: You cannot change the name of an topic.

Target object

A WebSphere MQ messaging provider topic at the specific scope.

Required parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider topic.

Optional parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider topic.

Note the behavior of this command on the **-customProperties** parameter.

-customProperties
This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider topic implementation. Typically, custom properties are used to set attributes of the topic that are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:
- In Jython: [*name value*]
- In Jacl: {*name value*}

New name/value pairs are added to the existing set of custom properties using the following rules:
- If the existing set of properties does not contain a property with the same name as that supplied as part of a modify command, the supplied property is added to the set of custom properties, unless the custom property has no value specified, when it is disregarded.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, and the modify command also specifies a value for the property, the existing value is replaced by the supplied value.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, but the modify command does not specify a value for the property, the property with the same name is deleted from the existing set of custom properties.

Example

- Using Jython:

```
wsadmin>AdminTask.modifyWMQTopic("t1(cells/L3A3316Node04Cell/
nodes/L3A3316Node05|resources.xml#MQTopic_1204538835312)", ["-priority 7"])
t1(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#
MQTopic_1204538835312)
```

- Using Jacl:

```
wsadmin>$AdminTask modifyWMQTopic
t1(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#
MQTopic_1204538835312) {-priority 7}
t1(cells/L3A3316Node04Cell/nodes/L3A3316Node05|resources.xml#
MQTopic_1204538835312)
```

showWMQTopic command

Use the showWMQTopic command to display information about a specific WebSphere MQ messaging provider topic.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the showWMQTopic command to display all the parameters, and their values, associated with a particular WebSphere MQ messaging provider topic.

Target object

A WebSphere MQ messaging provider topic at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQTopics("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
topic1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)

wsadmin>AdminTask.showWMQTopic("topic1(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#MQTopic_1098737234986)")
{specifiedExpiry=0, priority=APPLICATION_DEFINED, decimalEncoding=Normal,
baseTopicName=topic1, name=topic1, readAhead=NO, brokerPubQmgr=null,
readAheadClose=DELIVERCURRENT, brokerPubQueue=SYSTEM.BROKER.DEFAULT.STREAM,
ccsid=1208, integerEncoding=Normal, useNativeEncoding=true,
wildcardFormat=charWildcards, expiry=APP, specifiedPriority=0,
jndiName=jms/t1, sendAsync=NO,
brokerDurSubQueue=SYSTEM.JMS.D.SUBSCRIBER.QUEUE, brokerCCDurSubQueue=null,
description=null, targetClient=JMS, floatingPointEncoding=IEEENormal,
persistence=APP, brokerVersion=V1}
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQTopics
```

```
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
topic1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)

wsadmin>$AdminTask showWMQTopic
topic1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQTopic_1098737234986)
{specifiedExpiry=0, priority=APPLICATION_DEFINED, decimalEncoding=Normal,
baseTopicName=topic1, name=topic1, readAhead=NO, brokerPubQmgr=null,
readAheadClose=DELIVERCURRENT, brokerPubQueue=SYSTEM.BROKER.DEFAULT.STREAM,
ccsid=1208, integerEncoding=Normal, useNativeEncoding=true,
wildcardFormat=charWildcards, expiry=APP, specifiedPriority=0,
jndiName=jms/t1, sendAsync=NO,
brokerDurSubQueue=SYSTEM.JMS.D.SUBSCRIBER.QUEUE, brokerCCDurSubQueue=null,
description=null, targetClient=JMS, floatingPointEncoding=IEEENormal,
persistence=APP, brokerVersion=V1}
```

manageWMQ command

Use the manageWMQ command to manage the settings of the WebSphere MQ resource adapter that is installed at a particular scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the manageWMQ command to manage the settings associated with the WebSphere MQ resource adapter that is installed at a particular scope.

You can use the manageWMQ command to manage the native library path and query the metadata of the specified WebSphere MQ resource adapter.

Target object

A WebSphere MQ resource adapter.

Required parameters

None.

Optional parameters

-nativePath
This parameter specifies the path to the WebSphere MQ messaging provider native libraries that are used by the WebSphere MQ resource adapter to establish a bindings mode connection to the queue manager. This parameter can be specified on a WebSphere MQ adapter at any scope.

-query
This parameter provides information about the level of WebSphere MQ resource adapter that is used by the WebSphere MQ messaging provider. This parameter can be specified on a WebSphere MQ resource adapter at any scope.

-disableWMQ
This parameter specifies whether or not to disable WebSphere MQ functionality at the scope of the specified resource adapter, and at all scopes beneath it.

The value of this parameter must be true or false.

The default value is false.

In a single server environment this parameter is valid only at the server scope. In a network deployment environment this parameter is valid at all scopes. The affect of setting this parameter to true depends on the scope at which you set it:

- For a cell scoped WebSphere MQ resource adapter, all WebSphere MQ functionality on all application servers in the cell is disabled.
- For a node scoped WebSphere MQ resource adapter, all WebSphere MQ functionality on all application servers that are part of that node is disabled.
- For a cluster scoped WebSphere MQ resource adapter, all WebSphere MQ functionality on all application servers in that cluster are disabled.
- For a server scoped WebSphere MQ resource adapter, all WebSphere MQ functionality in that particular application server is disabled.

In all cases, all affected processes must be restarted for the changes to take effect.

The value of the parameter at a higher scope takes precedence over the value at a lower scope. For example, if you set the parameter to false at the server scope but a higher (for example, cell) scoped WebSphere MQ messaging provider has the parameter set to true, the value at the cell scope takes precedence and WebSphere MQ functionality is therefore disabled in all application servers in the cell, regardless of the parameter value at the server scope.

-maxConnections
This parameter specifies the maximum number of connections to a WebSphere MQ queue manager.

The default value is 10.

For further information, see Configuration of the ResourceAdapter object in the WebSphere MQ information center.

-connectionConcurrency
This parameter specifies the maximum number of message-driven beans that can be supplied by each connection.

The default value is 5.

For further information, see Configuration of the ResourceAdapter object in the WebSphere MQ information center.

-reconnectionRetryCount
This parameter specifies the maximum number of attempts made by a WebSphere MQ messaging provider activation specification to reconnect to a WebSphere MQ queue manager if a connection fails.

The default value is 5.

For further information, see Configuration of the ResourceAdapter object in the WebSphere MQ information center.

-reconnectionRetryInterval
This parameter specifies the time, in milliseconds, that a WebSphere MQ messaging provider activation specification waits before making another attempt to reconnect to a WebSphere MQ queue manager.

The default value is 300000.

For further information, see Configuration of the ResourceAdapter object in the WebSphere MQ information center.

-customProperties
This parameter specifies custom properties to be passed to the WebSphere MQ resource adapter. Typically, custom properties are used to set attributes of the WebSphere MQ resource adapter that are not directly supported through the WebSphere Application Server administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:
- In Jython: [*name value*]
- In Jacl: {*name value*}

New name/value pairs are added to the existing set of custom properties using the following rules:
- If the existing set of properties does not contain a property with the same name as that which is supplied, the supplied property is added to the set of custom properties, unless the custom property has no value specified, when it is disregarded.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, and the modify command also specifies a value for the property, the existing value is replaced by the supplied value.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, but the modify command does not specify a value for the property, the property with the same name is deleted from the existing set of custom properties.

Examples

The following example shows how to enable inbound JCA message delivery on the z/OS platform.

- Using Jython:

```
wsadmin>AdminTask.manageWMQ("WebSphere MQ Resource Adapter
(cells/L3A3316Node04Cell/nodes/L3A3316Node05/servers/server1|resources.xml#
J2CResourceAdapter_1201601803796)", ["-enableInbound true"])
```

- Using Jacl:

```
wsadmin>$AdminTask manageWMQ "WebSphere MQ Resource Adapter
(cells/L3A3316Node04Cell/nodes/L3A3316Node05/servers/server1|resources.xml#
J2CResourceAdapter_1201601803796)" {-enableInbound true}
```

The following example sets the value of **-maxConnections** to 100 and adds a custom property with name `name1` and value `value1`.

- Using Jython:

```
AdminTask.manageWMQ("WebSphere MQ Resource Adapter(
cells/L3A3316Node01Cell|resources.xml#J2CResourceAdapter_1284547647859)",
["-maxConnections 100 -customProperties [[name1 value1]]"])
```

- Using Jacl:

```
wsadmin>$AdminTask manageWMQ "WebSphere MQ Resource Adapter(
cells/L3A3316Node01Cell|resources.xml#J2CResourceAdapter_1284547647859)"
{-maxConnections 100 -customProperties {{name1 value1}}}
```

showWMQ command

Use the showWMQ command to show the settings which can be set by the manageWMQ command.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the showWMQ command to display all the parameters, and their values that can be set by the manageWMQ command. These settings are either related to the WebSphere MQ resource adapter or the WebSphere MQ messaging provider. The command also shows custom properties which are set on the WebSphere MQ resource adapter.

The following settings are expected:

- maxConnections
- connectionConcurrency
- reconnectionRetryCount
- reconnectionRetryInterval
- nativePath
- enableInbound
- disableWMQ (this parameter is only visible when setting it is valid, see the description of the manageWMQ command for more information)

For a description of these settings, see “manageWMQ command” on page 913.

Any other setting that is shown is either a custom property or a property that is not appropriate in WebSphere Application Server.

Target object

A WebSphere MQ resource adapter.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminTask.showWMQ("WebSphere MQ Resource Adapter
(cells/L3A3316Node01Cell|resources.xml#J2CResourceAdapter_1284547647859)")
'{name1=value1, logWriterEnabled=true, maxConnections=100, startupReconnectionRetryCount=500,
connectionConcurrency=123, reconnectionRetryCount=3, traceEnabled=false,
reconnectionRetryInterval=4, nativePath=[], startupReconnectionRetryInterval=600, traceLevel=3}'
```

- Using Jacl:

```
wsadmin>$AdminTask showWMQ "WebSphere MQ Resource Adapter(
cells/L3A3316Node01Cell|resources.xml#J2CResourceAdapter_1284547647859)"
{name1=value1, logWriterEnabled=true, maxConnections=100, startupReconnectionRetryCount=500,
connectionConcurrency=123, reconnectionRetryCount=3, traceEnabled=false,
reconnectionRetryInterval=4, nativePath=[], startupReconnectionRetryInterval=600, traceLevel=3}
```

migrateWMQMLP command

Use the migrateWMQMLP command to migrate a WebSphere MQ message listener port definition to an activation specification definition.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the migrateWMQMLP command to migrate a WebSphere MQ message listener port definition to an activation specification definition. After the activation specification has been created, you can delete the listener port.

Note that the **Maximum retries** listener port setting is not migrated to the new activation specification as there is no exact equivalent.

When you are migrating a listener port associated with a message-driven bean (MDB) that has the subscriptionDurability activation configuration property set to Durable, and that MDB already has an active durable subscription, the durable subscription is not migrated. This is because listener ports and WebSphere MQ activation specifications use incompatible forms of subscription name. As a result there can be two active durable subscriptions subscribed to the relevant topic for the same MDB. As part of the migration process, you must delete the old durable subscription that was associated with the listener port and manually clean up any messages associated with it. For information on how do to this see the WebSphere MQ and WebSphere Message Broker information centres.

Target object

The message listener port to be migrated.

Required parameters

-asName
The name of the activation specification to be created.

-asJNDIName
The JNDI name of the activation specification to be created.

-asScope
The type of scope at which to create the activation specification (server, node, cluster or cell). Note that the cluster option is only supported when the server that contains the message listener port is part of a cluster. If not specified this defaults to server. The scopes specified are relative to the message listener port, so node is the node of the server that contains the message listener port.

Optional parameters

None.

The following example shows how to migrate a message listener port to an activation specification.

- Using Jython:

```
wsadmin>AdminConfig.list("ListenerPort")
lp1(cells/L3A3316Node09Cell/nodes/L3A3316Node10/servers/server1|
server.xml#ListenerPort_1211265363796)

wsadmin>AdminTask.migrateWMQMLP("lp1(cells/L3A3316Node09Cell/nodes/
L3A3316Node10/servers/server1|server.xml#ListenerPort_1211265363796)",
 ["-asName migratedFromLP -asJNDIName jms/as1 -asScope node"])
migratedFromLP(cells/L3A3316Node09Cell/nodes/L3A3316Node10|
resources.xml#J2CActivationSpec_1211265679078)
```

- Using Jacl:

```
wsadmin>$AdminConfig list ListenerPort
lp1(cells/L3A3316Node09Cell/nodes/L3A3316Node10/servers/server1|
server.xml#ListenerPort_1211265363796)

wsadmin>$AdminTask migrateWMQMLP
lp1(cells/L3A3316Node09Cell/nodes/L3A3316Node10/servers/server1|
```

```
server.xml#ListenerPort_1211265363796)
{-asName migratedFromLP -asJNDIName jms/as1 -asScope node}
migratedFromLP(cells/L3A3316Node09Cell/nodes/L3A3316Node10|
resources.xml#J2CActivationSpec_1211265679078)
```

createWMQQueue command

Use the createWMQQueue command to create a queue type destination for the WebSphere MQ messaging provider at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the createWMQQueue command to create a WebSphere MQ messaging provider queue type destination at a specific scope.

You cannot create a WebSphere MQ messaging provider queue type destination under either of the following conditions:

- A WebSphere MQ messaging provider queue type destination already exists with the same name, at the same scope.
- The JNDI name clashes with another entry in WebSphere Application Server JNDI.

Target object

The scope of the WebSphere MQ messaging provider at which the WebSphere MQ messaging provider queue type destination is to be created.

Required parameters

-name
The administrative name assigned to this WebSphere MQ messaging provider queue type destination.

-jndiName
The name used to bind this object into WebSphere Application Server JNDI.

Optional parameters

-description
An administrative description assigned to the queue type destination.

-queueName
The name of the WebSphere MQ queue to use to store messages for the WebSphere MQ messaging provider queue type destination definition.

-qmgr
The queue manager that hosts the WebSphere MQ queue.

-persistence
This parameter determines the level of persistence used to store messages sent to this destination.

Enter one of the following case-sensitive values:

- APP
- QDEF
- PERS
- NON
- HIGH

The default value is APP.

-priority
The priority level to assign to messages sent to this destination.

Enter one of the following case-sensitive values:

- APP
- QDEF

or enter a positive integer in the range 0 to 9 (inclusive).

The default value is APP.

-expiry
The length of time after which messages sent to this destination expire and are dealt with according to their disposition options.

Enter one of the following case-sensitive values:

- APP
- UNLIM

or enter any positive integer.

The default value is APP.

-ccsid
The coded character set identifier (CCSID).

The value of this parameter must be a positive integer or blank. See the “WebSphere MQ messaging provider queue and topic advanced properties settings” on page 844 for more details.

The default value is 1208.

Leaving this field empty indicates that the default value must be used.

-useNativeEncoding
This parameter specifies whether to use native encoding or not. It can take a value true or false.

If it is set to true, the values of the **-integerEncoding**, **-decimalEncoding** and **-floatingPointEncoding** attributes are ignored.

If it is set to false, the encoding is specified by the **-integerEncoding**, **-decimalEncoding** and **-floatingPointEncoding** attributes.

-integerEncoding
The integer encoding setting for this queue.

Enter one of the following case-sensitive values: Normal or Reversed.

The default value is Normal.

-decimalEncoding
The decimal encoding setting for this queue.

Enter one of the following case-sensitive values: Normal or Reversed.

The default value is Normal.

-floatingPointEncoding
The floating point encoding setting for this queue.

Enter one of the following case-sensitive values: IEEENormal, IEEEReversed, z/OS

The default value is IEEENormal.

-useRFH2
This parameter determines whether an RFH version 2 header is appended to messages sent to this destination.

Enter one of the following case-sensitive values: true or false.

The default value is true.

-sendAsync
This parameter determines whether messages can be sent to this destination without queue manager acknowledging that they have arrived.

Enter one of the following case-sensitive values: YES, NO or QDEF.

The default value is YES.

-readAhead
This parameter determines whether messages for non-persistent consumers can be read ahead and cached.

Enter one of the following case-sensitive values: YES, NO or QDEF.

The default value is YES.

-readAheadClose

Enter one of the following case-sensitive values: YES, NO or QDEF.

The default value is YES.

-customProperties
This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider queue type destination implementation. Typically, custom properties are used to set attributes of the queue type destination that are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:
- In Jython: [*name value*]
- In Jacl: {*name value*}

The following example creates a WebSphere MQ messaging provider queue type destination.

- Using Jython:

```
wsadmin>AdminTask.createWMQQueue("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)", ["-name queue1 -jndiName jms/queues/Q1
 -queueName APP1.QUEUE1"])
queue1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)
```

- Using Jacl:

```
wsadmin>$AdminTask createWMQQueue
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
{-name queue1 -jndiName jms/queues/Q1 -queueName APP1.QUEUE1}
queue1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)
```

deleteWMQQueue command

Use this command to delete a WebSphere MQ messaging provider queue type destination at a specific scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the deleteWMQQueue command to delete a WebSphere MQ messaging provider queue type destination defined at the scope at which the command is issued.

Note: The WebSphere MQ messaging provider queue type destination definition is deleted from WebSphere Application Server JNDI at the specific scope. Any messages present on the underlying WebSphere MQ queue are not affected.

Target object

A WebSphere MQ messaging provider queue type destination at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQQueues("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
unwantedQueue(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)

wsadmin>AdminTask.deleteWMQQueue("unwantedQueue(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|resources.xml#MQQueue_1098737234986)")
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQQueues
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
unwantedQueue(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)

wsadmin>$AdminTask deleteWMQQueue
unwantedQueue(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)
```

listWMQQueues command

Use the listWMQQueues command to list WebSphere MQ messaging provider queue type destinations.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the listWMQQueues command to list all of the WebSphere MQ messaging provider queue type destinations defined at the scope at which the command is issued.

Target object

WebSphere MQ messaging provider queue type destinations at the specific scope.

Required parameters

None.

Optional parameters

-type
If this parameter it is omitted all connection factories are shown at the appropriate scope.

Enter one of the following case-sensitive values: CF, QCF or TCF.Enter CF to list only common connection factories, QCF to list only queue connection factories or **TCF** to list only topic connection factories.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQQueues("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQQueues
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.
```

modifyWMQQueue command

Use the modifyWMQQueue command to change certain parameters of a WebSphere MQ messaging provider queue type destination.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the modifyWMQQueue command to modify a WebSphere MQ messaging provider queue type destination defined at the scope at which the command is issued.

Target object

A WebSphere MQ messaging provider queue type destination at the specific scope.

Required parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider queue type destination.

Optional parameters

The parameters for this command are identical to those used to create a WebSphere MQ messaging provider queue type destination.

Note the behavior of this command on the **-customProperties** parameter.

-customProperties

This parameter specifies custom properties to be passed to the WebSphere MQ messaging provider queue type destination implementation. Typically, custom properties are used to set attributes of the queue type destination which are not directly supported through the WebSphere administration interfaces.

Each custom property is specified using name and value table step parameters. Since these are table steps, the order of the two parameters is fixed, so you must always specify the name first and the value second:

- In Jython: [*name value*]
- In Jacl: {*name value*}

New name/value pairs are added to the existing set of custom properties using the following rules:

- If the existing set of properties does not contain a property with the same name as that supplied as part of a modify command, the supplied property is added to the set of custom properties, unless the custom property has no value specified, when it is disregarded.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, and the modify command also specifies a value for the property, the existing value is replaced by the supplied value.
- If the existing set of properties contains a property with the same name as that supplied as part of a modify command, but the modify command does not specify a value for the property, the property with the same name is deleted from the existing set of custom properties.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>AdminTask.listWMQQueues("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)

wsadmin>AdminTask.modifyWMQQueue("jmsq2(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#
MQQueue_1098737234986)", ["-ccsid 500"])
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQQueues
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)

wsadmin>$AdminTask modifyWMQQueue
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986) {-ccsid 500}
jmsq2(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.
```

showWMQQueue command

Use the showWMQQueue command to display information about a specific WebSphere MQ messaging provider queue type destination.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available WebSphere MQ messaging provider administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WMQAdminCommands')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

Use the showWMQQueue command to display all the parameters, and their values, associated with a particular WebSphere MQ messaging provider queue type destination.

Target object

A WebSphere MQ messaging provider queue type destination at the specific scope.

Required parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminConfig.getid("/Node:9994GKCNode01")
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
```

```
wsadmin>AdminTask.listWMQQueues("9994GKCNode01(cells/9994GKCNode01Cell/
nodes/9994GKCNode01|node.xml#Node_1)")
ncq1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)

wsadmin>AdminTask.showWMQQueue("ncq1(cells/9994GKCNode01Cell/nodes/
9994GKCNode01|resources.xml#MQQueue_1098737234986)")
{expiry=0, priority=9, decimalEncoding=Normal, queueName=TARGETQ, name=q1,
readAhead=YES, readAheadClose=DELIVERALL,  ccsid=1208,
useNativeEncoding=true, integerEncoding=Normal, specifiedPriority=0,
jndiName=jms/q1, sendAsync=YES,  qmgr=null, description=null,
targetClient=JMS, floatingPointEncoding=IEEENormal, persistence=APP}
```

- Using Jacl:

```
wsadmin>$AdminConfig getid /Node:9994GKCNode01
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)

wsadmin>$AdminTask listWMQQueues
9994GKCNode01(cells/9994GKCNode01Cell/nodes/9994GKCNode01|node.xml#Node_1)
ncq1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)

wsadmin>$AdminTask showWMQQueue
ncq1(cells/9994GKCNode01Cell/nodes/9994GKCNode01|resources.xml#
MQQueue_1098737234986)
{expiry=0, priority=9, decimalEncoding=Normal, queueName=TARGETQ, name=q1,
readAhead=YES, readAheadClose=DELIVERALL,  ccsid=1208,
useNativeEncoding=true, integerEncoding=Normal, specifiedPriority=0,
jndiName=jms/q1, sendAsync=YES,  qmgr=null, description=null,
targetClient=JMS, floatingPointEncoding=IEEENormal, persistence=APP}
```

Mapping of administrative console panel names to command names and WebSphere MQ names

Use these tables to relate the names used in the administrative console panels to the names in the commands and the names used by WebSphere MQ.

The following tables list property mappings. Use these tables in conjunction with the WebSphere MQ information center to get more information on a particular property.

Note that not every WebSphere Application Server panel or command property maps to a WebSphere MQ property.

Table 62. WebSphere MQ Activation Specification property mappings. The first column of this table shows the property names that appear on a WebSphere Application Server administrative console panel, and the second column shows the corresponding WebSphere Application Server command names. The third column shows the WebSphere MQ activation specification properties that the administrative console panel names and command names map on to.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ activation specification property
Basic panel		
Queue manager	qmgrName	queueManager
Queue manager (when used with CCDT)	ccdtQmgrName	queueManager
Transport	wmqTransportType	transportType
Host name	qmgrHostname	hostName
Port	qmgrPortNumber	port
Server connection channel	qmgrSvrconnChannel	channel
Destination JNDI name	destinationJndiName	destination

Table 62. WebSphere MQ Activation Specification property mappings (continued). The first column of this table shows the property names that appear on a WebSphere Application Server administrative console panel, and the second column shows the corresponding WebSphere Application Server command names. The third column shows the WebSphere MQ activation specification properties that the administrative console panel names and command names map on to.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ activation specification property
Message selector	messageSelector	messageSelector
Destination type	destinationType	destinationType
Durable subscription	subscriptionDurability	subscriptionDurability
Subscription name	subscriptionName	subscriptionName
Client ID	clientID	clientID
Provider version	providerVersion	providerVersion
Client channel definition table URL	ccdtUrl	ccdtURL
Allow cloned durable subscriptions	clonedSubs	cloneSupport
Connection name list	connectionNameList	connectionNameList
Advanced properties panel		
Compress message headers	compressHeaders	headerCompression
Compression algorithm for message payloads	compressPayload	messageCompression
Retain messages, even if no matching consumer is available	msgRetention	messageRetention
Rescan interval	rescanInterval	rescanInterval
Maximum server sessions	maxPoolSize	maxPoolDepth
Start timeout	startTimeout	startTimeout
Server session pool timeout	poolTimeout	poolTimeout
Coded character set identifier	ccsid	CCSID
Fail JMS method calls if the queue manager is quiescing	failIfQuiescing	failIfQuiesce
Broker properties panel		
Broker control queue	brokerCtrlQueue	brokerControlQueue
Broker durable subscriber connection consumer queue	brokerCCDurSubQueue	brokerCCDurSubQueue
Broker subscriber queue	brokerSubQueue	brokerSubQueue
Broker connection consumer subscription queue	brokerCCSubQueue	brokerCCSubQueue
Version	brokerVersion	brokerVersion
Specify where message selection occurs	msgSelection	messageSelection
Subscription store	subStore	subscriptionStore
Durable subscription state refresh interval	stateRefreshInt	statusRefreshInterval
Subscription cleanup level	cleanupLevel	cleanupLevel
Subscription cleanup interval	cleanupInterval	cleanupInterval
Subscription wildcard format	wildcardFormat	wildcardFormat
Optimize for sparse subscription patterns	sparseSubs	sparseSubscriptions
Broker queue manager	brokerQmgr	brokerQueueManager
Client transport properties panel		
Certificate revocation list	sslCrl	sslCertStores
Peer name	sslPeerName	sslPeerName
Reset count	sslResetCount	sslResetCount
Receive exits	rcvExit	receiveExit

Table 62. WebSphere MQ Activation Specification property mappings (continued). The first column of this table shows the property names that appear on a WebSphere Application Server administrative console panel, and the second column shows the corresponding WebSphere Application Server command names. The third column shows the WebSphere MQ activation specification properties that the administrative console panel names and command names map on to.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ activation specification property
Receive exit initialization data	rcvExitInitData	receiveExitInit
Send exits	sendExit	sendExit
Send exit initialization data	sendExitInitData	sendExitInit
Security exit	secExit	securityExit
Security exit initialization data	secExitInitData	securityExitInit
Other properties that are only on commands		
	localAddress	localAddress

Table 63. WebSphere MQ connection factory property mappings. This table shows how the connection factory property names that appear on a WebSphere Application Server administrative console panel, and the corresponding WebSphere Application Server command names map to WebSphere MQ JMS Admin short and long names.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ JMS Admin short name	WebSphere MQ JMS Admin long name
Basic panel			
Description	description	DESC	DESCRIPTION
Queue manager	qmgrName	QMGR	QMANAGER
Queue manager (when used with client channel definition table (CCDT))	ccdtQmgrName	QMGR	QMANAGER
Transport	wmqTransportType	TRAN	TRANSPORT
Host name	qmgrHostname	HOST	HOSTNAME
Port	qmgrPortNumber	PORT	PORT
Server connection channel	qmgrSvrconnChannel	CHAN	CHANNEL
Client ID	clientID	CID	CLIENTID
Allow cloned durable subscriptions	clonedSubs	CLS	CLONESUPP
Provider version	providerVersion	PVER	PROVIDERVERSION
Client channel definition table URL	ccdtUrl	CCDT	CCDTURL
Connection name list	connectionNameList	CNLIST	CONNECTIONNAMELIST
Advanced properties panel			
Client reconnect options	clientReconnectOptions	CROPT	CLIENTRECONNECT OPTIONS
Client reconnect timeout	clientReconnectTimeout	CRT	CLIENTRECONNECT TIMEOUT
Compress message headers	compressHeaders	HC	COMPHDR
Compression algorithm for message payloads	compressPayload	MC	COMPMSG
WebSphere MQ model queue name	modelQueue	TM	TEMPMODEL
Temporary queue prefix	tempQueuePrefix	TQP	TEMPQPREFIX
Temporary topic prefix	tempTopicPrefix	TTP	TEMPTOPICPREFIX
Retain messages, even if no matching consumer is available	msgRetention	MRET	MSGRETENTION
Polling interval	pollingInterval	PINT	POLLINGINT

Table 63. WebSphere MQ connection factory property mappings (continued). This table shows how the connection factory property names that appear on a WebSphere Application Server administrative console panel, and the corresponding WebSphere Application Server command names map to WebSphere MQ JMS Admin short and long names.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ JMS Admin short name	WebSphere MQ JMS Admin long name
Rescan interval	rescanInterval	RINT	RESCANINT
Maximum batch size	maxBatchSize	MBS	MSGBATCHSZ
Coded character set identifier	ccsid	CCS	CCSID
Append an RFH version 2 header to reply messages	replyWithRFH2	TCM	TARGCLIENTMATCHING
Fail JMS method calls if the queue manager is quiescing	failIfQuiescing	FIQ	FAILIFQUIESCE
Broker properties panel			
Broker control queue	brokerCtrlQueue	BCON	BROKERCONQ
Broker publication queue	brokerPubQueue	BPUB	BROKERPUBQ
Broker subscriber queue	brokerSubQueue	BSUB	BROKERSUBQ
Broker connection consumer subscription queue	brokerCCSubQueue	CCSUB	BROKERCCSUBQ
Version	brokerVersion	BVER	BROKERVER
Specify where message selection occurs	msgSelection	MSEL	MSGSELECTION
Subscription store	subStore	SS	SUBSTORE
Durable subscription state refresh interval	stateRefreshInt	SRI	STATREFRESHINT
Subscription cleanup level	cleanupLevel	CL	CLEANUP
Subscription cleanup interval	cleanupInterval	CLINT	CLEANUPINT
Subscription wildcard format	wildcardFormat	WCFMT	WILDCARDFORMAT
Publish acknowledgement window	pubAckInterval	PAI	PUBACKINT
Optimize for sparse subscription patterns	sparseSubs	SSUBS	SPARSESUBS
Broker queue manager	brokerQmgr	BQM	BROKERQMGR
Client transport properties panel			
Certificate revocation list	sslCrl	CRL	SSLCRL
Peer name	sslPeerName	SPEER	SSLPEERNAME
Reset count	sslResetCount	SRC	SSLRESETCOUNT
Receive exits	rcvExit	RCX	RECEXIT
Receive exit initialization data	rcvExitInitData	RCXI	RECEXITINIT
Send exits	sendExit	SDX	SENDEXIT
Send exit initialization data	sendExitInitData	SDXI	SENDEXITINIT
Security exit	secExit	SCX	SECEXIT
Security exit initialization data	secExitInitData	SCXI	SECEXITINIT
Other properties that are only on commands			
	localAddress	LA	LOCALADDRESS

Table 64. WebSphere MQ Queue property mappings. This table shows how the queue property names that appear on a WebSphere Application Server administrative console panel, and the corresponding WebSphere Application Server command names map to WebSphere MQ JMS Admin short and long names.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ JMS Admin short name	WebSphere MQ JMS Admin Long Name
Basic panel			
Description	description	DESC	DESCRIPTION
Queue name	queueName	QU	QUEUE
Queue manager or Queue-sharing group name	qmgr	QMGR	QMANAGER
Advanced properties panel			
Persistence	persistence	PER	PERSISTENCE
Priority	priority	PRI	PRIORITY
Expiry	expiry	EXP	EXPIRY
Coded character set identifier	ccsid	CCS	CCSID
Native encoding/Integer encoding/Decimal encoding/Floating point encoding	useNativeEncoding/ integerEncoding/decimalEncoding/ floatingPointEncoding	ENC	ENCODING
Append RFH version 2 headers to messages sent to this destination	useRFH2	TC	TARGCLIENT
Message body	messageBody	MBODY	MSGBODY
ReplyTo destination style	replyToDestinationStyle	RTOST	REPLYTOSTYLE
Asynchronously send messages to the queue manager	sendAsync	PAALD	PUTASYNCALLOWED
Read ahead, and cache, non-persistent messages for consumers	readAhead	RAALD	READAHEADALLOWED
Read ahead consumer close method	readAheadClose	RACP	READAHEADCLOSE POLICY
MQMD read enabled	mqmdReadEnabled	MDR	MDREAD
MQMD write enabled	mqmdWriteEnabled	MDW	MDWRITE
MQMD message context	mqmdMessageContext	MDCTX	MDMSGCTX

Table 65. WebSphere MQ topic property mappings. This table shows how the topic property names that appear on a WebSphere Application Server administrative console panel, and the corresponding WebSphere Application Server command names map to WebSphere MQ JMS Admin short and long names.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ JMS Admin short name	WebSphere MQ JMS Admin long name
Basic panel			
Description	description	DESC	DESCRIPTION
Topic name	topicName	TOP	TOPIC
Broker durable subscription queue	brokerDurSubQueue	BDSUB	BROKERDURSUBQ
Broker durable subscriber connection consumer queue	brokerCCDurSubQueue	CCDSUB	BROKERCCDURSUBQ
Broker publication queue	brokerPubQueue	BPUB	BROKERPUBQ
Broker publication queue manager	brokerPubQmgr	BQM	BROKERPUBQMGR
Advanced properties panel			
Persistence	persistence	PER	PERSISTENCE
Priority	priority	PRI	PRIORITY
Expiry	expiry	EXP	EXPIRY

Table 65. WebSphere MQ topic property mappings (continued). This table shows how the topic property names that appear on a WebSphere Application Server administrative console panel, and the corresponding WebSphere Application Server command names map to WebSphere MQ JMS Admin short and long names.

Name on WebSphere Application Server administrative console panel	WebSphere Application Server command name	WebSphere MQ JMS Admin short name	WebSphere MQ JMS Admin long name
Coded character set identifier	ccsid	CCS	CCSID
Native encoding/Integer encoding/Decimal encoding/Floating point encoding	useNativeEncoding/ integerEncoding/decimalEncoding/ floatingPointEncoding	ENC	ENCODING
Append RFH version 2 headers to messages sent to this destination	useRFH2	TC	TARGCLIENT
Message body	messageBody	MBODY	MSGBODY
ReplyTo destination style	replyToDestinationStyle	RTOST	REPLYTOSTYLE
Asynchronously send messages to the queue manager	sendAsync	PAALD	PUTASYNCALLOWED
Read ahead, and cache, non-persistent messages for consumers	readAhead	RAALD	READAHEADALLOWED
Read ahead consumer close method	readAheadClose	RACP	READAHEADCLOSE POLICY
MQMD read enabled	mqmdReadEnabled	MDR	MDREAD
MQMD write enabled	mqmdWriteEnabled	MDW	MDWRITE
MQMD message context	mqmdMessageContext	MDCTX	MDMSGCTX
Other properties that are only on commands			
	wildcardFormat	WCFMT	WILDCARDFORMAT
	brokerVersion	BVER	BROKERVER

Chapter 14. Managing messaging with a third-party or (deprecated) V5 default messaging provider

For messaging between application servers, most requirements are best met by either the default messaging provider or the WebSphere MQ messaging provider. However, you can instead use a third-party messaging provider (that is, use another company's product as the provider). You might want to do this, for example, if you have existing investments. For backwards compatibility with earlier releases, there is also support for the V5 default messaging provider.

Before you begin

If you are not sure which provider combination is best suited to your needs, see Types of messaging providers.

About this task

Enterprise applications in WebSphere Application Server can use asynchronous messaging through services based on Java Message Service (JMS) messaging providers and their related messaging systems. These messaging providers conform to the JMS Version 1.1 specification.

The choice of provider depends on what your JMS application needs to do, and on other factors relating to your business environment and planned changes to that environment.

Procedure

- Choose a third-party messaging provider.

  To administer a third-party messaging provider, you use either the resource adaptor (for a Java EE Connector Architecture (JCA) 1.5-compliant messaging provider) or the client (for a non-JCA messaging provider) that is supplied by the third party. You use the WebSphere Application Server administrative console to administer the activation specifications, connection factories and destinations that are within WebSphere Application Server, but you cannot use the administrative console to administer the JMS provider itself, or any of its resources that are outside of WebSphere Application Server.

  To use message-driven beans, third-party messaging providers must either provide an inbound JCA 1.5-compliant resource adapter, or (for non-JCA messaging providers) include Application Server Facility (ASF), an optional feature that is part of the JMS Version 1.1 specification.

  To work with a third-party provider, choose one of the following options:

  1. Manage messaging with a third-party JCA 1.5-compliant messaging provider.
  2. Manage messaging with a third-party non-JCA messaging provider.

- Choose the (deprecated) V5 default messaging provider.

  This deprecated provider is identical to the WebSphere Application Server Version 5 default provider. Only the name has changed. It provides backwards compatibility that enables WebSphere Application Server Version 6 or later applications to connect to WebSphere Application Server Version 5 resources in a mixed cell. It also allows WebSphere Application Server Version 5 applications to connect to WebSphere Application Server Version 6 or later resources in a mixed cell. To configure and manage messaging to interoperate with WebSphere Application Server Version 5, see “Maintaining (deprecated) Version 5 default messaging resources” on page 959.

Managing messaging with a third-party JCA 1.5-compliant messaging provider

You can configure WebSphere Application Server to use a third-party JCA 1.5-compliant messaging provider. You might want to do this, for example, if you have existing investments.

Before you begin

For messaging between application servers, perhaps with some interaction with a WebSphere MQ system, you can use the default messaging provider. To integrate WebSphere Application Server messaging into a predominately WebSphere MQ network, you can use the WebSphere MQ messaging provider. You can also use a third-party messaging provider as described in this topic. To choose the provider that is best suited to your needs, see Choosing a messaging provider.

About this task

A third-party JCA 1.5-compliant messaging provider takes the form of a resource adapter that you install in WebSphere Application Server. You use the administrative console to administer the activation specifications (for message-driven beans) and other J2C administered objects for the provider.

Procedure

- Install the third-party JCA 1.5-compliant messaging provider.

  Install the resource adapter for the third-party provider, as described in “Installing a resource adapter archive” on page 180.
- Configure an activation specification for a third-party JCA resource adapter.
- Configure an administered object for a third-party JCA resource adapter.

Configuring an activation specification for a third-party JCA resource adapter

You can configure an activation specification that is used to deploy message-driven beans with a Java™ Connector Architecture (JCA), also called J2EE Connector (J2C), resource adapter that is not included as part of the WebSphere Application Server.

Before you begin

For guidance about when to configure your message-driven beans to work with listener ports rather than activation specifications. see Message-driven beans, activation specifications, and listener ports.

This task assumes that you have installed the resource adapter for the third-party provider, as described in “Installing a resource adapter archive” on page 180.

About this task

Use this task if you want to use a message-driven bean as a listener on a JCA resource adapter other than the default messaging provider or the WebSphere MQ messaging provider.

Procedure

1. Start the administrative console.
2. Display the resource adapter. In the navigation pane, click **Resources** > **Resource Adapters** > *adapter_name*. This displays in the content pane a table of properties for the resource adapter, including links to the types of JCA resource that it provides.
3. Optional: Change the **Scope** setting to the scope level at which the activation specification is to be visible to applications, according to your needs.
4. In the content pane, under the Activation specifications heading, click **J2C Activation Specifications**. This lists any existing activation specifications for the resource adapter.
5. Display the properties of the activation specification.

   If you want to display an existing activation specification, click one of the names listed.

Alternatively, if you want to create a new activation specification, click **New**, then specify the following required properties:

Name Type the name by which the activation specification is known for administrative purposes. The JNDI name is automatically generated based on the value for the Name property.

Message listener type
Select the message listener type that this activation specification instance should support. This list is based on the deployment descriptor of the resource adapter.

Depending on the resource adapter, there can be additional required properties that you must supply. To provide values for these properties, click **Custom properties**. When creating a new activation specification, you might have to click **Apply** before this custom property selection is available.

6. Specify properties for the activation specification, according to your needs.
7. Click **OK**.
8. Save your changes to the master configuration.

J2C Activation Specifications collection

This page contains a list of Java 2 Connector (J2C) activation specifications for a resource adapter configuration and is used to create new J2C activation specifications, to select J2C activation specifications for configuration changes, or to delete J2C activation specifications.

Activation specification definitions and classes are provided by a resource adapter when it is installed. Using this information, the administrator can create and configure J2C activation specifications with JNDI names that are then available for applications to use. The resource adapter uses a J2C activation specification to configure a specific endpoint instance. Each application configuring one or more endpoints must specify the resource adapter that sends messages to the endpoint. The application must use the activation specification to provide the configuration properties related to the processing of the inbound messages.

The following guidelines show which scenarios use activation specifications or listener ports:

- If you are using Java 2 Platform, Enterprise Edition (J2EE) 1.2 and EJB 1.1 with WebSphere Application Server v4, message driven beans (MDB) are not used so you do not need listener ports or activation specifications. WebSphere Application Server v4 uses message beans, but these are not MDBs or Enterprise JavaBeans (EJB).
- If you are using J2EE 1.3 and EJB 2.0 with WebSphere Application Server v5, you must use listener ports. The MDBs are JMS MDBs that implement MessageListener, and there is no JCA support. WebSphere Application Server v5 uses listener ports to associate MDB classes with their JMS destinations.
- If you are using J2EE 1.4 and EJB 2.1 with WebSphere Application Server v6, the decision depends on whether your JMS provider API is implemented with JCA. In J2EE 1.4, the JMS 1.1 API can now be implemented with the JCA 1.5 API. If so, your MDB is a JMS MDB that is implemented as a connector MDB, and must therefore be configured with an activation specification. If not, this is the same JMS situation as for J2EE 1.3, and you must configure this EJB 2.1 MDB in the same way as you would configure an EJB 2.0 MDB, which in WebSphere Application Server is to use a listener port.

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **J2C activation specifications**.
- **Resources** > **Resource Adapters** > **J2C activation specifications**.

Name:

Specifies the display name of the J2C activation specification instance.

A string with no spaces meant to be a meaningful text identifier for the J2C activation specification.

Data type String

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name for the J2C activation specification instance.

Data type String

Scope:

Specifies the scope of the resource adapter that supports this activation specification. Only applications that are installed within this scope can use this activation specification.

Provider:

Specifies the resource adapter that encapsulates the appropriate classes for this activation specification.

Description:

A free-form text string to describe the J2C activation specification instance.

Data type String

Message Listener Type:

The Message Listener Type that is used by this activation specification.

The list of available classes is provided by the resource adapter.

Data type String

J2C Activation Specifications settings:

Use this page to specify the settings for a Java 2 Connector (J2C) activation specification.

The resource adapter uses a J2C activation specification to configure a specific endpoint instance. Each application configuring one or more endpoints must specify the resource adapter that sends messages to the endpoint. The application must use the activation specification to provide the configuration properties related to the processing of the inbound messages.

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **J2C activation specifications** > ***activation_specification***.
- **Resources** > **Resource Adapters** > **J2C activation specifications** > ***activation_specification***.

Scope:

Specifies the scope of the resource adapter that supports this activation specification. Only applications that are installed within this scope can use this activation specification.

Provider:

Specifies the resource adapter that encapsulates the appropriate classes for this activation specification.

For new objects, available resource adapters are listed in a drop-down list. After you create the activation specification, the field is a read only text field.

Data type Drop-down list or text

Name:

Specifies the display name of the J2C activation specification instance.

A string with no spaces meant to be a meaningful text identifier for the J2C activation specification. Name is required

Data type String

JNDI name:

Specifies the Java Naming and Directory Interface (JNDI) name for the J2C activation specification instance.

The JNDI name is required. If you do not specify one, it is created from the Name field. If not specified, the JNDI name defaults to *eis/[name]*

Data type String

Description:

A free-form text string to describe the J2C activation specification instance.

Data type String

Authentication alias:

This optional field is used to bind the J2C activation specification to an authentication alias (configured through the security JAAS screens).

This alias is used to access a user name and password that are set on the configured J2C activation specification. This field is meaningful only if the J2C activation specification that you are configuring has a UserName and Password field.

If you have defined security domains in the application server, you can click **Browse...** to select a J2C authentication alias for the resource that you are configuring. Security domains allow you to isolate J2C authentication aliases between servers. The tree view is useful in determining the security domain to which an alias belongs, and the tree view can help you determine the servers that is able to access each authentication alias. The tree view is tailored for each resource, so domains and aliases are hidden when you cannot use them.

The Browse button is only accessible if at least one security domain is defined and assigned a scope that is applicable to the resource that is being edited. Additionally, that security domain must contain at least one JAAS J2C Authentication alias.

Data type Drop-down list

Message Listener Type:

The Message Listener Type used by this activation specification.

For new objects, the list of available classes is provided by the resource adapter in a drop-down list. After you create the activation specification, the field is a read only text field.

Data type Drop-down list or text

Destination JNDIName:

The destination JNDIName field appears only when a message of type javax.jms.Destination with name *Destination* is received.

Configuring an administered object for a third-party JCA resource adapter

You can configure an administered object for a Java™ Connector Architecture (JCA), also called J2EE Connector (J2C), resource adapter that is not included as part of the WebSphere Application Server.

Before you begin

This task assumes that you have installed the resource adapter for the third-party provider, as described in “Installing a resource adapter archive” on page 180.

About this task

Use this task if you want to configure a JCA administered object for a JCA resource adapter other than the default messaging provider or the WebSphere MQ messaging provider.

Procedure

1. Start the administrative console.
2. Display the resource adapter. In the navigation pane, click **Resources** > **Resource Adapters** > *adapter_name*. This displays in the content pane a table of properties for the resource adapter, including links to the types of JCA resource that it provides.
3. Optional: Change the **Scope** setting to the scope level at which the administered object is to be visible to applications, according to your needs.
4. In the content pane, under the Additional Properties heading, click **J2C Administered Objects**. This lists any existing administered objects for the resource adapter.
5. Display the properties of the administered object.

   If you want to display an existing administered object, click one of the names listed.

   Alternatively, if you want to create a new administered object, click **New**, then specify the following required properties:

   Name Type the name by which the administered object is known for administrative purposes. The JNDI name is automatically generated based on the value for the Name property.

   Administered object class
   Select the administered object class that this instance should support. This list is based on the deployment descriptor of the resource adapter.

   Depending on the resource adapter, there can be additional required properties that you must supply. To provide values for these properties, click **Custom properties**. When creating a new administered object, you might have to click **Apply** before this custom property selection is available.
6. Specify properties for the administered object, according to your needs.

7. Click **OK**.
8. Save your changes to the master configuration.

J2C Administered Objects collection

This page contains a list of Java 2 Connector Architecture (JCA) administered objects for a resource adapter configuration. This page is also used to create JCA administered objects, select J2C administered objects for configuration changes, or to delete J2C administered objects.

Administered object definitions and classes are provided by a resource adapter when you install it. Using this information, the administrator can create and configure J2C administered objects with JNDI names that are then available for applications to use. Some messaging styles might need applications to use special administered objects for sending and synchronously receiving messages through connection objects with messaging style-specific APIs. Administered objects can also be used to perform transformations on an asynchronously received message in a message provider-specific way. Administered objects can be accessed by a component by using either a resource environment reference or a message destination reference (preferred).

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **J2C administered objects**
- **Resources** > **Resource Adapters** > **Resource adapters** > **J2C administered objects**

Name:

Specifies display name assigned to this administered object.

Data type String

JNDI Name:

Specifies the JNDI name of the administered object.

Data type String

Scope:

Specifies the scope of the resource adapter that supports this administered object. Only applications that are installed within this scope can use this object.

Provider:

Specifies the resource adapter that encapsulates the appropriate classes for this administrative object.

For new objects, available resource adapters are listed in a drop-down list. After you create the administered object, the field is a read only text field.

Data type Drop-down list or text

Description:

Specifies a description for the administered object.

Data type String

Administered object class:

Specifies the Administered Object class that is associated with this J2C administered object. This class must be one that is provided by the resource adapter.

Data type String

J2C Administered Object settings:

Use this page to specify the settings for a Java 2 Connector (J2C) administered object.

Administered object definitions and classes are provided by a resource adapter when you install it. Using this information, the administrator can create and configure J2C administered objects with Java Naming and Directory Interface (JNDI) names that are then available for applications to use. Some messaging styles may need applications to use special administered objects for sending and synchronously receiving messages (through connection objects using messaging style specific APIs). It is also possible that administered objects may be used to perform transformations on an asynchronously received message in a message provider-specific way. Administered objects can be accessed by a component by using either a resource environment reference or a message destination reference (preferred).

You can access this administrative console page in one of two ways:

- **Resources** > **Resource Adapters** > **Resource adapters** > ***resource_adapter*** > **J2C administered objects** > ***J2C_administered_object***
- **Resources** > **Resource Adapters** > **Resource adapters** > **J2C administered objects** > ***J2C_administered_object***

Scope:

Specifies the scope of the resource adapter that supports this administered object. Only applications that are installed within this scope can use this object.

Provider:

Specifies the resource adapter that encapsulates the appropriate classes for this administrative object.

For new objects, a list of available resource adapters appears in a drop-down list. After you create the administered object, the field is a read only text field.

Data type Drop-down list or text

Name:

Specifies the name of the J2C administered object instance.

A string with no spaces meant to be a meaningful text identifier for the administered object. This name is required.

Data type String

JNDI name:

Specifies the JNDI name that this administered object is bound under.

The JNDI name is required. If you do not specify one, it is created from the Name field. If not specified, the JNDI name defaults to *eis/[name]*

Data type String

Description:

Specifies a text description of the J2C administered object instance.

Data type String

Administered object class:

For new objects, the list of available classes is provided by the resource adapter in a drop-down list. You can only select classes from this list.

After you create the administered object, you cannot modify the administered object class; it is read only.

Data type Class name

Managing messaging with a third-party non-JCA messaging provider

You can configure WebSphere Application Server to use a third-party non-JCA messaging provider. You might want to do this, for example, if you have existing investments. You can configure any third-party non-JCA messaging provider that supports the JMS Version 1.1 unified connection factory.

Before you begin

For messaging between application servers, perhaps with some interaction with a WebSphere MQ system, you can use the default messaging provider. To integrate WebSphere Application Server messaging into a predominately WebSphere MQ network, you can use the WebSphere MQ messaging provider. You can also use a third-party messaging provider as described in this topic. To choose the provider that is best suited to your needs, see Choosing a messaging provider.

To work with message-driven beans, the third-party non-JCA messaging provider must include Application Server Facility (ASF), an optional feature that is part of the JMS Version 1.1 specification.

About this task

To administer a third-party non-JCA messaging provider, you use the client that is supplied by the third party. You use the administrative console to administer the connection factories and destinations that are within WebSphere Application Server, but you cannot use the administrative console to administer the JMS provider itself, or any of its resources that are outside of WebSphere Application Server.

Procedure

- Define a third-party non-JCA messaging provider.
- List JMS resources for a third-party non-JCA messaging provider.
- Configure JMS resources for a third-party non-JCA messaging provider.

Defining a third-party non-JCA messaging provider

Use this task to define a third-party non-JCA messaging provider to WebSphere Application Server. You might want to do this, for example, if you have existing investments.

Before you begin

Before you define a third-party non-JCA messaging provider, you might want to check whether your requirement can be met by the default messaging provider or the WebSphere MQ messaging provider that are supplied with WebSphere Application Server. To choose the provider that is best suited to your needs, see Choosing a messaging provider.

To work with message-driven beans, the third-party non-JCA messaging provider must include Application Server Facility (ASF), an optional feature that is part of the JMS Version 1.1 specification.

About this task

You can configure any third-party non-JCA messaging provider that supports the JMS Version 1.1 unified connection factory.

To administer a third-party non-JCA messaging provider, you use the client that is supplied by the third party. You use the administrative console to administer the connection factories and destinations that are within WebSphere Application Server, but you cannot use the administrative console to administer the JMS provider itself, or any of its resources that are outside of WebSphere Application Server.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Resources** > **JMS->JMS providers**. The existing messaging providers are displayed, including the default messaging provider and the WebSphere MQ messaging provider.
3. To define a new third-party non-JCA messaging provider, click **New** in the content pane. Otherwise, to change the definition of an existing messaging provider, click the name of the provider.
4. Specify the following required properties. You can specify other properties, as described in a later step.

   Name The name by which this messaging provider is known for administrative purposes within WebSphere Application Server.

   External initial context factory
   The Java classname of the initial context factory for the JMS provider.

   External provider URL
   The JMS provider URL for external JNDI lookups.
5. Optional: Click **Apply**. This enables you to specify additional properties.
6. Optional: Specify other properties for the messaging provider.

   Under Additional Properties, you can use the **Custom Properties** link to specify custom properties for your initial context factory, in the form of standard **javax.naming** properties.
7. Click **OK**.
8. Save the changes to the master configuration.
9. To have the changed configuration take effect, stop then restart the application server.

What to do next

You can now configure JMS resources for your messaging provider, as described in “Configuring JMS resources for a third-party non-JCA messaging provider” on page 952.

JMS provider settings

Use this panel to view the configuration properties of a selected JMS provider.

To view this page, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->JMS providers**. This displays a list of JMS providers in the content pane. For each JMS provider in the list, the entry indicates the *scope* level at which JMS

resource definitions are visible to applications. You can create the same type of JMS provider at different Scope settings, to offer JMS resources at different levels of visibility to applications.

2. (optional) If you want to manage JMS resources that are defined at a different scope setting, change the **Scope** setting to the required level.
3. In the Providers column of the list displayed, click the name of a JMS provider.

If you want to browse or change JMS resources of the JMS provider, click the link for the type of resource under Additional Properties. For more information about the administrative console panels for the types of JMS resources, see the related topics.

For the default messaging provider (which is based on service integration technologies) and the WebSphere MQ messaging provider, the scope, name, and description properties are displayed for information only. You cannot change these properties.

For a third-party non-JCA provider that you have defined yourself, the properties of that provider are displayed.

Scope:

The level (cell, node, or server level) at which this resource definition is available.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes. For more information about the scope setting, see Scope settings.

Name:

The name by which the JMS provider is known for administrative purposes.

Data type

String

Default

- Default messaging provider.

  For JMS resources to be provided by a service integration bus, as part of WebSphere Application Server.
- *My JMSprovider*

  For JMS resources to be provided by a third-party JMS provider that you specify, rather than by the default messaging provider or the WebSphere MQ messaging provider that are available as part of WebSphere Application Server. You assign the name, for example “My JMSprovider”, when you define the third-party JMS provider to WebSphere Application Server. You must also have installed and configured the third-party JMS provider before applications can use its JMS resources.
- WebSphere MQ messaging provider

  For JMS resources to be provided by WebSphere MQ. You must have installed and configured WebSphere MQ before applications can use its JMS resources.
- V5 default messaging provider.

  For JMS resources to be provided by a WebSphere Application Server Version 5 node.

Description:

A description of the JMS provider, for administrative purposes within WebSphere Application Server.

Data type String

Classpath:

A list of paths or JAR file names that together form the location for the JMS provider classes. Each class path entry is on a separate line (separated by using the Enter key) and must not contain path separator characters (such as ';' or ': '). Class paths can contain variable (symbolic) names to be substituted by using a variable map. Check your driver installation notes for specific JAR file names that are required.

Note: This property is only available for third-party messaging providers.

Data type String

Native library path:

An optional path to any native libraries (*.dll, *.so). Each native path entry is on a separate line (separated by using the Enter key) and must not contain path separator characters (such as ';' or ': '). Native paths can contain variable (symbolic) names to be substituted by using a variable map.

Note: This property is only available for the WebSphere MQ messaging provider and third-party messaging providers.

Data type String

Update resource adapter:

This button can be used to update the WebSphere MQ resource adapter that provides the function made available by the WebSphere MQ messaging provider. This button must only be used as directed by a member of IBM service, otherwise it may result in the use of an unsupported level of the WebSphere MQ resource adapter.

Normally the WebSphere MQ resource adapter is automatically updated by applying WebSphere Application Server fix packs. It is important to note that use of the **Update resource adapter** button prevents these automatic updates from happening for future fix packs for any node on which the button is used. If, in the future, you require the WebSphere MQ resource adapter used by the node to receive updates when a fix pack is applied then you must re-establish the recommended resource adapter configuration. For more information see “Maintaining the WebSphere MQ resource adapter” on page 724.

Note: This property is only available for the WebSphere MQ messaging provider.

Data type Button

External initial context factory:

The Java classname of the initial context factory for the JMS provider.

For example, for an LDAP service provider the value has the form: com.sun.jndi.ldap.LdapCtxFactory.

Note: This property is only available for third-party messaging providers.

Data type String
Default Null

External provider URL:

The JMS provider URL for external JNDI lookups.

For example, an LDAP URL for a messaging provider has the form: `ldap://hostname.company.com/contextName`.

Note: This property is only available for third-party messaging providers.

Data type	String
Default	Null

Disable WebSphere MQ:

This check box is only valid for the WebSphere MQ messaging provider. When selected, this check box disables all WebSphere MQ functionality on affected application servers. Note that you must restart the affected application server processes for this change to take effect.

In a single server environment this check box is only available on the WebSphere MQ messaging provider panel where the scope is set to server, and has the effect of disabling all WebSphere MQ functionality in that application server.

In a network deployment environment this check box is available on all WebSphere MQ messaging provider panels. The effect of selecting the check box depends on the scope at which you select it:

- At the cell scope, all WebSphere MQ functionality is disabled on all application servers in the cell.
- At the node scope, all WebSphere MQ functionality is disabled on all application servers that are part of that node.
- At the cluster scope, all WebSphere MQ functionality is disabled on all application servers in that cluster.
- At the server scope, all WebSphere MQ functionality is disabled in that particular application server.

The value of the check box at a higher scope takes precedence over the value at a lower scope. For example, if you do not select the check box for a WebSphere MQ messaging provider at the server scope, but do select it for a WebSphere MQ messaging provider at a higher scope (such as the cell scope), the value of the check box at the cell scope takes precedence and WebSphere MQ functionality is therefore disabled in all application servers in the cell, regardless of whether the check box is selected at the server scope.

An informational message indicating that WebSphere MQ has been disabled is added to all WebSphere MQ messaging provider panels that are at affected scopes, but this message does not appear on those panels where the check box is selected. In a single server environment this informational message is only displayed after a server restart is performed. In a network deployment environment the informational message is displayed immediately.

For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Data type	Check box
Default	Not selected

Additional properties:

Connection factories

A connection factory is used to create connections to the associated JMS provider. These connection factories can be used for accessing JMS Queue and JMS Topic destinations.

Queue connection factories
A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for point-to-point messaging.

Topic connection factories
A topic connection factory is used to create connections to the associated JMS provider of JMS topic destinations, for publish and subscribe messaging.

Queues
A JMS queue is used as a destination for point-to-point messaging.

Topics
A JMS topic is used as a destination for publish/subscribe messaging.

Activation specifications
A JMS activation specification is associated with one or more message-driven beans and provides configuration necessary for them to receive messages.

Resource adapter properties
These properties are used to configure the WebSphere MQ resource adapter, which is used by the WebSphere MQ messaging provider. In particular most of these settings affect the behavior of WebSphere MQ messaging provider activation specifications.

Listing JMS resources for a third-party non-JCA messaging provider

Use the WebSphere Application Server administrative console to list JMS resources for a third party non-JCA messaging provider.

Before you begin

This topic assumes that you have defined a third-party non-JCA messaging provider.

About this task

You use the WebSphere Application Server administrative console to list JMS resources, if you want to view, modify or delete any of the following resources:

- Unified connection factories
- Queue connection factories
- Topic connection factories
- Queues
- Topics

When you use the administrative console to locate these resources, two different navigation pathways are available:

- **Provider-centric navigation** lets you view all providers, or just those for a specified scope, then navigate to a specific resource for a specific provider. This is the traditional way of navigating to a resource when you know which provider supports it. Any navigation that starts with **Resources** > **JMS->JMS providers** is provider-centric.
- **Resource-centric navigation** lets you view all resources of a specified type, then navigate to a resource. This is useful if you want to find a resource, but you do not know which provider supports it (you can list all resources of a given type across all scopes, for all providers, in a single panel). Any navigation that follows the pattern **Resources** > **JMS** > ***resource_type*** is resource-centric, where *resource_type* is one of the resource types previously listed.

You can use either of these navigation pathways to locate JMS resources of any type.

Procedure

- Use provider-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console.
  2. In the navigation pane, click **Resources** > **JMS->JMS providers**.

     The JMS providers collection panel is displayed. This lists all currently configured messaging providers across all scopes (you can modify the scope if required).
  3. Select the required JMS provider.

     The settings panel for this provider is displayed. The configuration tab contains a set of links to all the JMS resources owned by this provider.
  4. Click the link for a JMS resource type. For example, click **Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories for this provider.
  5. Select the required queue connection factory.
- Use resource-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console..
  2. In the navigation pane, click **Resources** > **JMS->Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories across all messaging providers.
  3. Select the required queue connection factory.

Results

You can now view and work with the resource properties.

JMS providers collection

Use this panel to list JMS providers, or to select a JMS provider to view or change its configuration properties.

To view this administrative console page, click **Resources** > **JMS->JMS providers**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS providers that are available to WebSphere applications. For each JMS provider in the list, the entry indicates the *scope* level at which JMS resource definitions are visible to applications. You can create the same type of JMS provider at different Scope settings, to offer JMS resources at different levels of visibility to applications.

If you want to manage existing JMS resource definitions, or create a new JMS resource definition, you can select the name of one of the JMS providers in the list.

If you want to define a new third-party JMS provider (that is, a provider other than the default messaging provider or the WebSphere MQ messaging provider), select the Scope setting at which JMS resource definitions are to be visible for that provider, then click **New**.

Name The name by which this JMS provider is known for administrative purposes.

Description
A description of this JMS provider for administrative purposes.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Connection factory collection

A JMS connection factory is used to create connections to the associated messaging provider of JMS destinations, for both point-to-point and publish/subscribe messaging. Use connection factory administrative objects to manage JMS connection factories for the default messaging provider, the WebSphere MQ messaging provider or a third-party messaging provider.

In the administrative console, to view this page click **Resources** > **JMS->Connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS connection factory is used to create connections to JMS destinations. When an application needs a JMS connection, an instance can be created by the factory of the messaging provider named in the **Provider** column of the list.

This type of connection factory is for applications that use the JMS 1.1 domain-independent interfaces (referred to as the “common interfaces” in the JMS specification).

This type of JMS connection factory can also be used by the domain-specific (queue and topic) interfaces, as used in JMS 1.0.2, so applications can still use those interfaces without the need for you to create a domain-specific connection factory, such as a queue connection factory.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The display name of each connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each connection factory instance.

Provider
The messaging provider that supports each connection factory instance. This is the default messaging provider (service integration), the WebSphere MQ messaging provider or a third-party messaging provider.

Description
An optional description of each connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Queue connection factory collection

A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for point-to-point messaging.

In the administrative console, to view this page click **Resources** > **JMS->Queue connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS queue connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS queue connection factory is used to create connections to JMS destinations. When an application needs a JMS queue connection, an instance can be created by the factory for the JMS provider that is named in the **Provider** column of the list.

This type of connection factory is for applications that use the JMS 1.0.2 queue-specific interfaces.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The display name of each queue connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each queue connection factory instance.

Provider
The messaging provider that supports each queue connection factory instance.

Description
An optional description of each queue connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Topic connection factory collection

A JMS topic connection factory is used to create connections to the associated messaging provider of JMS topic destinations, for publish and subscribe messaging.

In the administrative console, to view this page click **Resources** > **JMS->Topic connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS topic connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS connection factory is used to create connections to JMS destinations. When an application needs a JMS connection, an instance can be created by the factory for the JMS provider that is named in the Provider column of the list.

This type of connection factory is for applications that use the JMS 1.0.2 topic-specific interfaces.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each topic connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each topic connection factory instance.

Provider
The messaging provider that supports each topic connection factory instance.

Description
An optional description of each topic connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Queue collection

A JMS queue destination is used for point-to-point messaging. Use this panel to create or delete queue destinations, or to select a queue destination to view or change its configuration properties.

In the administrative console, to view this page click **Resources** > **JMS->Queues**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS queue destinations that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

Use a queue destination to manage JMS queues for the JMS provider that is named in the **Provider** column of the list. Connections to the queue are created by a unified connection factory or queue connection factory for that JMS provider.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each queue destination instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each queue destination instance.

Provider
The messaging provider that supports each queue destination instance.

Description
An optional description of each queue destination instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Topic collection

A JMS topic destination is used for publish and subscribe messaging. Use this panel to create or delete topic destinations, or to select a topic destination to view or change its configuration properties.

In the administrative console, to view this page click **Resources** > **JMS->Topics**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS topic destinations that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

Use a topic destination to manage JMS topics for the JMS provider that is named in the **Provider** column of the list. Connections to the topic are created by a unified connection factory or topic connection factory for that JMS provider.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each topic destination instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each topic destination instance.

Provider
The messaging provider that supports each topic destination instance.

Description
An optional description of each topic destination instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Configuring JMS resources for a third-party non-JCA messaging provider

Use this task to configure the JMS connection factories and destinations for a third-party non-JCA messaging provider.

Before you begin

You only have to complete these tasks if your WebSphere Application Server environment uses a third-party non-JCA messaging provider to support enterprise applications that use JMS.

Before you can configure resources for your third-party non-JCA messaging provider, you must have defined your messaging provider.

About this task

To configure JMS resources for a third-party non-JCA messaging provider, complete the following tasks:

Procedure

- Configure a JMS connection factory for a third-party non-JCA messaging provider.
- Configure a JMS destination for a third-party non-JCA messaging provider.

Configuring a JMS connection factory for a third-party non-JCA messaging provider

Use this task to view or change the properties of a JMS connection factory for use with a third-party non-JCA messaging provider.

About this task

To view or change the configuration of a JMS connection factory for use with a third-party non-JCA messaging provider, use the administrative console to complete the following steps:

Procedure

1. Display the third-party non-JCA messaging provider. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. Select the third-party non-JCA messaging provider for which you want to configure a connection factory.
3. Optional: Select the **Scope** setting corresponding to the scope of the connection factories that you want to view or change.
4. In the content pane, under Additional Properties, click **Connection factories** This displays a table listing any existing JMS connection factories, with a summary of their properties.
5. To browse or change an existing JMS connection factory, click its name in the list. Otherwise, to create a new connection factory, complete the following steps:
   a. Click **New** in the content pane.
   b. Specify the following required properties. You can specify other properties, as described in a later step.

      Name The name by which this JMS connection factory is known for administrative purposes within IBM WebSphere Application Server.

      Type Select whether the connection factory is for JMS queues (QUEUE) or JMS topics (TOPIC).

      JNDI Name
      The JNDI name that is used to bind the JMS connection factory into the WebSphere Application Server namespace.

      External JNDI Name
      The JNDI name that is used to bind the JMS connection factory into the namespace of the messaging provider.

   c. Click **Apply**. This defines the JMS connection factory to WebSphere Application Server, and enables you to browse or change additional properties.
6. Optional: Change properties for the JMS connection factory, according to your needs.
7. Click **OK**.
8. Save any changes to the master configuration.
9. To have the changed configuration take effect, stop then restart the application server.

Third-party JMS connection factory settings:

Use this panel to browse or change the configuration properties of a JMS connection factory configured for use with a third-party non-JCA messaging provider. These configuration properties control how connections are created to the JMS destinations on the provider.

A JMS connection factory is used to create connections to JMS destinations. The JMS connection factory is created by the associated JMS provider.

To view this page, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->JMS providers**. The JMS providers collection panel is displayed. This lists all currently configured messaging providers across all scopes (you can modify the scope if required).
2. Click the name of the third-party non-JCA messaging provider.
3. Under Additional Properties, click **Connection factories**.
4. Click the name of the JMS connection factory that you want to work with.

Scope:

Specifies the level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the messaging provider settings panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Name:

The name by which this JMS connection factory is known for administrative purposes within IBM WebSphere Application Server. The name must be unique within the associated messaging provider.

Data type String

Type:

Whether this connection factory is for creating JMS queue destinations or JMS topic destinations.

Select one of the following options:

QUEUE
A JMS queue connection factory for point-to-point messaging.

TOPIC
A JMS topic connection factory for publish/subscribe messaging.

JNDI name:

The JNDI name that is used to bind the connection factory into the WebSphere Application Server namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Description:

A description of this connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String
Default Null

Category:

A category used to classify or group this connection factory, for your IBM WebSphere Application Server administrative records.

Data type String

External JNDI name:

The JNDI name that is used to bind the connection factory into the namespace of the third-party messaging provider.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Component-managed Authentication Alias:

This alias specifies a user ID and password to be used to authenticate connection to a JMS provider for application-managed authentication.

This property provides a list of the J2C authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate the creation of a new connection to the JMS provider.

If you have enabled administrative security for WebSphere Application Server, select the alias that specifies the user ID and password used to authenticate the creation of a new connection to the JMS provider. The use of this alias depends on the resource authentication (res-auth) setting declared in the connection factory resource reference of the deployment descriptors for an application component.

Container-managed Authentication Alias:

This alias specifies a user ID and password to be used to authenticate connection to a JMS provider for container-managed authentication.

This property provides a list of the J2C authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate the creation of a new connection to the JMS provider.

If you have enabled administrative security for WebSphere Application Server, select the alias that specifies the user ID and password used to authenticate the creation of a new connection to the JMS provider. The use of this alias depends on the resource authentication (res-auth) setting declared in the connection factory resource reference of the deployment descriptors for an application component.

Mapping-Configuration Alias:

The module used to map authentication aliases.

This field provides a list of the modules that have been configured on the **Global Security** > **JAAS Configuration** > **Application Logins Configuration** property. For more information about the mapping configurations, see Java Authentication and Authorization service configuration entry settings.

Data type Enum

Default Null

Range

ClientContainer
The client container maps authentication aliases.

WSLogin
The WSLogin module maps authentication aliases.

DefaultPrincipalMapping
The JAAS configuration maps an authentication alias to its userid and password.

Connection pool:

Specifies an optional set of connection pool settings.

Connection pool properties are common to all J2C connectors.

The application server pools connections and sessions with the messaging provider to improve performance. You have to configure the connection and session pool properties appropriately for your applications, otherwise you may not get the connection and session behavior that you want.

Change the size of the connection pool if concurrent server-side access to the JMS resource exceeds the default value. The size of the connection pool is set on a per queue or topic basis.

Session pool:

An optional set of session pool settings.

This link provides a panel of optional connection pool properties, common to all J2C connectors.

The application server pools connections and sessions with the messaging provider to improve performance. You have to configure the connection and session pool properties appropriately for your applications, otherwise you may not get the connection and session behavior that you want.

Custom properties:

An optional set of name and value pairs for custom properties passed to the messaging provider.

Configuring a JMS destination for a third-party non-JCA messaging provider

Use this task to browse or change the properties of a JMS destination for use with a third-party non-JCA messaging provider.

Before you begin

Before starting this task, you should have defined the messaging provider to WebSphere Application Server.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Resources** > **JMS->JMS providers**.
3. Click the name of the third-party non-JCA messaging provider.
4. In the content pane, under Additional Properties, click **Queues** for point-to-point messaging or **Topics** for publish/subscribe messaging. This displays a table listing any existing JMS destinations, with a summary of their properties.
5. To browse or change an existing JMS destination, click its name in the list. Otherwise, to create a new destination, complete the following steps:
   a. Click **New** in the content pane.
   b. Specify the following required properties. You can specify other properties, as described in a later step.

      Name The name by which this JMS destination is known for administrative purposes within WebSphere Application Server.

      Type Select whether the destination is for JMS queues (QUEUE) or JMS topics (TOPIC).

      JNDI Name
      The JNDI name that is used to bind the JMS destination into the WebSphere Application Server namespace.

      External JNDI Name
      The JNDI name that is used to bind the JMS destination into the namespace of the messaging provider.

   c. Click **Apply**. This defines the JMS destination to WebSphere Application Server, and enables you to browse or change additional properties.
6. Optional: Change properties for the JMS destination, according to your needs.
7. Click **OK**.
8. Save any changes to the master configuration.
9. To have the changed configuration take effect, stop then restart the application server.

Third-party JMS destination settings:

Use this panel to browse or change the configuration properties of the selected JMS destination for use with an associated third-party non-JCA messaging provider.

A JMS destination is used to configure the properties of a JMS destination for the associated third-party non-JCA messaging provider. Connections to the JMS destination are created by the associated JMS connection factory.

To navigate to this panel, complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->JMS providers**. The JMS providers collection panel is displayed. This lists all currently configured messaging providers across all scopes (you can modify the scope if required).
2. Click the name of the third-party non-JCA messaging provider.

3. Under Additional Properties, click **Queues** for point-to-point messaging or **Topics** for publish/subscribe messaging.
4. Click the name of the JMS destination that you want to work with.

Scope:

Specifies the level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the messaging provider settings panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Name:

The name by which the queue is known for administrative purposes within WebSphere Application Server.

Data type String

Type:

Whether this JMS destination is a queue (for point-to-point) or topic (for publish/subscribe).

Select one of the following options:
Queue
A JMS queue destination for point-to-point messaging.
Topic A JMS topic destination for publish/subscribe messaging.

JNDI name:

The JNDI name that is used to bind the queue into the application server namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Description:

A description of the queue, for administrative purposes

Data type String

Category:

A category used to classify or group this queue, for your WebSphere Application Server administrative records.

Data type String

External JNDI name:

The JNDI name that is used to bind the queue into the application server namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Maintaining (deprecated) Version 5 default messaging resources

You can use the WebSphere Application Server administrative console to manage the JMS connection factories and destinations for WebSphere Application Server Version 5.1 applications. JMS connection factories and destinations for the WebSphere Application Server Version 5.1 default messaging provider are not the same as those for the default messaging provider in WebSphere Application Server Version 6 and later. Such JMS resources are maintained as *V5 default messaging* resources.

About this task

V5 default messaging provides a JMS transport to a messaging engine of a service integration bus that supports the default messaging provider in later versions of the product. The messaging engine emulates the service of a JMS server running on WebSphere Application Server Version 5.1.

Procedure

- List Version 5 default messaging resources.
- Configure Version 5 default messaging resources.

JMS provider settings

Use this panel to view the configuration properties of a selected JMS provider.

To view this page, use the administrative console to complete the following steps:

1. In the navigation pane, click **Resources** > **JMS->JMS providers**. This displays a list of JMS providers in the content pane. For each JMS provider in the list, the entry indicates the *scope* level at which JMS resource definitions are visible to applications. You can create the same type of JMS provider at different Scope settings, to offer JMS resources at different levels of visibility to applications.
2. (optional) If you want to manage JMS resources that are defined at a different scope setting, change the **Scope** setting to the required level.
3. In the Providers column of the list displayed, click the name of a JMS provider.

If you want to browse or change JMS resources of the JMS provider, click the link for the type of resource under Additional Properties. For more information about the administrative console panels for the types of JMS resources, see the related topics.

For the default messaging provider (which is based on service integration technologies) and the WebSphere MQ messaging provider, the scope, name, and description properties are displayed for information only. You cannot change these properties.

For a third-party non-JCA provider that you have defined yourself, the properties of that provider are displayed.

Scope
The level (cell, node, or server level) at which this resource definition is available.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes. For more information about the scope setting, see Scope settings.

Name
The name by which the JMS provider is known for administrative purposes.

Data type String

Default

- Default messaging provider.

  For JMS resources to be provided by a service integration bus, as part of WebSphere Application Server.
- *My JMSprovider*

  For JMS resources to be provided by a third-party JMS provider that you specify, rather than by the default messaging provider or the WebSphere MQ messaging provider that are available as part of WebSphere Application Server. You assign the name, for example “My JMSprovider”, when you define the third-party JMS provider to WebSphere Application Server. You must also have installed and configured the third-party JMS provider before applications can use its JMS resources.
- WebSphere MQ messaging provider

  For JMS resources to be provided by WebSphere MQ. You must have installed and configured WebSphere MQ before applications can use its JMS resources.
- V5 default messaging provider.

  For JMS resources to be provided by a WebSphere Application Server Version 5 node.

Description
A description of the JMS provider, for administrative purposes within WebSphere Application Server.

Data type String

Classpath
A list of paths or JAR file names that together form the location for the JMS provider classes. Each class path entry is on a separate line (separated by using the Enter key) and must not contain path separator characters (such as ';' or ': '). Class paths can contain variable (symbolic) names to be substituted by using a variable map. Check your driver installation notes for specific JAR file names that are required.

Note: This property is only available for third-party messaging providers.

Data type String

Native library path

An optional path to any native libraries (*.dll, *.so). Each native path entry is on a separate line (separated by using the Enter key) and must not contain path separator characters (such as ';' or ': '). Native paths can contain variable (symbolic) names to be substituted by using a variable map.

Note: This property is only available for the WebSphere MQ messaging provider and third-party messaging providers.

Data type String

Update resource adapter

This button can be used to update the WebSphere MQ resource adapter that provides the function made available by the WebSphere MQ messaging provider. This button must only be used as directed by a member of IBM service, otherwise it may result in the use of an unsupported level of the WebSphere MQ resource adapter.

Normally the WebSphere MQ resource adapter is automatically updated by applying WebSphere Application Server fix packs. It is important to note that use of the **Update resource adapter** button prevents these automatic updates from happening for future fix packs for any node on which the button is used. If, in the future, you require the WebSphere MQ resource adapter used by the node to receive updates when a fix pack is applied then you must re-establish the recommended resource adapter configuration. For more information see “Maintaining the WebSphere MQ resource adapter” on page 724.

Note: This property is only available for the WebSphere MQ messaging provider.

Data type Button

External initial context factory

The Java classname of the initial context factory for the JMS provider.

For example, for an LDAP service provider the value has the form: com.sun.jndi.ldap.LdapCtxFactory.

Note: This property is only available for third-party messaging providers.

Data type String
Default Null

External provider URL

The JMS provider URL for external JNDI lookups.

For example, an LDAP URL for a messaging provider has the form: `ldap://hostname.company.com/contextName`.

Note: This property is only available for third-party messaging providers.

Data type String
Default Null

Disable WebSphere MQ

This check box is only valid for the WebSphere MQ messaging provider. When selected, this check box disables all WebSphere MQ functionality on affected application servers. Note that you must restart the affected application server processes for this change to take effect.

In a single server environment this check box is only available on the WebSphere MQ messaging provider panel where the scope is set to server, and has the effect of disabling all WebSphere MQ functionality in that application server.

In a network deployment environment this check box is available on all WebSphere MQ messaging provider panels. The effect of selecting the check box depends on the scope at which you select it:

- At the cell scope, all WebSphere MQ functionality is disabled on all application servers in the cell.
- At the node scope, all WebSphere MQ functionality is disabled on all application servers that are part of that node.
- At the cluster scope, all WebSphere MQ functionality is disabled on all application servers in that cluster.
- At the server scope, all WebSphere MQ functionality is disabled in that particular application server.

The value of the check box at a higher scope takes precedence over the value at a lower scope. For example, if you do not select the check box for a WebSphere MQ messaging provider at the server scope, but do select it for a WebSphere MQ messaging provider at a higher scope (such as the cell scope), the value of the check box at the cell scope takes precedence and WebSphere MQ functionality is therefore disabled in all application servers in the cell, regardless of whether the check box is selected at the server scope.

An informational message indicating that WebSphere MQ has been disabled is added to all WebSphere MQ messaging provider panels that are at affected scopes, but this message does not appear on those panels where the check box is selected. In a single server environment this informational message is only displayed after a server restart is performed. In a network deployment environment the informational message is displayed immediately.

For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Data type	Check box
Default	Not selected

Additional properties

Connection factories
A connection factory is used to create connections to the associated JMS provider. These connection factories can be used for accessing JMS Queue and JMS Topic destinations.

Queue connection factories
A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for point-to-point messaging.

Topic connection factories
A topic connection factory is used to create connections to the associated JMS provider of JMS topic destinations, for publish and subscribe messaging.

Queues
A JMS queue is used as a destination for point-to-point messaging.

Topics
A JMS topic is used as a destination for publish/subscribe messaging.

Activation specifications
A JMS activation specification is associated with one or more message-driven beans and provides configuration necessary for them to receive messages.

Resource adapter properties
These properties are used to configure the WebSphere MQ resource adapter, which is used by

the WebSphere MQ messaging provider. In particular most of these settings affect the behavior of WebSphere MQ messaging provider activation specifications.

Listing Version 5 default messaging resources

Use the WebSphere Application Server administrative console to list JMS resources for the V5 default messaging provider, for administrative purposes.

About this task

You use the WebSphere Application Server administrative console to list JMS resources, if you want to view, modify or delete any of the following resources:

- Queue connection factories
- Topic connection factories
- Queues
- Topics

When you use the administrative console to locate these resources, two different navigation pathways are available:

- **Provider-centric navigation** lets you view all providers, or just those for a specified scope, then navigate to a specific resource for a specific provider. This is the traditional way of navigating to a resource when you know which provider supports it. Any navigation that starts with **Resources** > **JMS->JMS providers** is provider-centric.
- **Resource-centric navigation** lets you view all resources of a specified type, then navigate to a resource. This is useful if you want to find a resource, but you do not know which provider supports it (you can list all resources of a given type across all scopes, for all providers, in a single panel). Any navigation that follows the pattern **Resources** > **JMS** > ***resource_type*** is resource-centric, where *resource_type* is one of the resource types previously listed.

You can use either of these navigation pathways to locate JMS resources of any type.

Procedure

- Use provider-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console.
  2. In the navigation pane, click **Resources** > **JMS->JMS providers**.

     The JMS providers collection panel is displayed. This lists all currently configured messaging providers across all scopes (you can modify the scope if required).
  3. Select the required JMS provider.

     The settings panel for this provider is displayed. The configuration tab contains a set of links to all the JMS resources owned by this provider.
  4. Click the link for a JMS resource type. For example, click **Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories for this provider.
  5. Select the required queue connection factory.
- Use resource-centric navigation, for example to navigate to a specified queue connection factory.
  1. Start the administrative console..
  2. In the navigation pane, click **Resources** > **JMS->Queue connection factories**.

     The queue connection factories collection panel is displayed. This panel lists all the queue connection factories across all messaging providers.
  3. Select the required queue connection factory.

Results

You can now view and work with the resource properties.

JMS providers collection

Use this panel to list JMS providers, or to select a JMS provider to view or change its configuration properties.

To view this administrative console page, click **Resources** > **JMS->JMS providers**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS providers that are available to WebSphere applications. For each JMS provider in the list, the entry indicates the *scope* level at which JMS resource definitions are visible to applications. You can create the same type of JMS provider at different Scope settings, to offer JMS resources at different levels of visibility to applications.

If you want to manage existing JMS resource definitions, or create a new JMS resource definition, you can select the name of one of the JMS providers in the list.

If you want to define a new third-party JMS provider (that is, a provider other than the default messaging provider or the WebSphere MQ messaging provider), select the Scope setting at which JMS resource definitions are to be visible for that provider, then click **New**.

Name The name by which this JMS provider is known for administrative purposes.

Description
A description of this JMS provider for administrative purposes.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Queue connection factory collection

A queue connection factory is used to create connections to the associated JMS provider of the JMS queue destinations, for point-to-point messaging.

In the administrative console, to view this page click **Resources** > **JMS->Queue connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS queue connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS queue connection factory is used to create connections to JMS destinations. When an application needs a JMS queue connection, an instance can be created by the factory for the JMS provider that is named in the **Provider** column of the list.

This type of connection factory is for applications that use the JMS 1.0.2 queue-specific interfaces.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The display name of each queue connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each queue connection factory instance.

Provider
The messaging provider that supports each queue connection factory instance.

Description
An optional description of each queue connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Topic connection factory collection

A JMS topic connection factory is used to create connections to the associated messaging provider of JMS topic destinations, for publish and subscribe messaging.

In the administrative console, to view this page click **Resources** > **JMS->Topic connection factories**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS topic connection factories that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

A JMS connection factory is used to create connections to JMS destinations. When an application needs a JMS connection, an instance can be created by the factory for the JMS provider that is named in the Provider column of the list.

This type of connection factory is for applications that use the JMS 1.0.2 topic-specific interfaces.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each topic connection factory instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each topic connection factory instance.

Provider
The messaging provider that supports each topic connection factory instance.

Description
An optional description of each topic connection factory instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Queue collection

A JMS queue destination is used for point-to-point messaging. Use this panel to create or delete queue destinations, or to select a queue destination to view or change its configuration properties.

In the administrative console, to view this page click **Resources** > **JMS->Queues**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS queue destinations that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

Use a queue destination to manage JMS queues for the JMS provider that is named in the **Provider** column of the list. Connections to the queue are created by a unified connection factory or queue connection factory for that JMS provider.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server

environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each queue destination instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each queue destination instance.

Provider
The messaging provider that supports each queue destination instance.

Description
An optional description of each queue destination instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Topic collection

A JMS topic destination is used for publish and subscribe messaging. Use this panel to create or delete topic destinations, or to select a topic destination to view or change its configuration properties.

In the administrative console, to view this page click **Resources** > **JMS->Topics**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

This page lists the JMS topic destinations that are available to WebSphere Application Server applications at the scope indicated by the **Scope** field.

Use a topic destination to manage JMS topics for the JMS provider that is named in the **Provider** column of the list. Connections to the topic are created by a unified connection factory or topic connection factory for that JMS provider.

If WebSphere MQ functionality has been disabled, an informational message indicating that WebSphere MQ has been disabled appears when the scope field is set to a scope which is the same as, or above, the scope at which WebSphere MQ has been disabled, or when the scope field is set to display all scopes. Note that this informational message is not displayed if you are viewing a provider specific collection that is not for WebSphere MQ (for example, the default messaging provider collection). In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

General properties

Name The display name of each topic destination instance.

JNDI name
The Java Naming and Directory Interface (JNDI) name of each topic destination instance.

Provider
The messaging provider that supports each topic destination instance.

Description
An optional description of each topic destination instance.

Scope The level to which this resource definition is visible; for example, the cell, node, cluster, or server level.

Buttons

New	Create a new JMS resource of this type.
Delete	Delete the selected items.

Configuring Version 5 default messaging resources

Use the following tasks to configure the JMS connection factories and destinations for WebSphere Application Server Version 5 applications.

About this task

You only have to complete these tasks if you have WebSphere Application Server Version 5 applications that use JMS resources provided by the default messaging provider. Such JMS resources are maintained as *V5 default messaging* resources.

Procedure

- Configure a connection for Version 5 default messaging
- Configure a queue connection factory for Version 5 default messaging
- Configure a topic connection factory for Version 5 default messaging
- Configure a queue for Version 5 default messaging
- Configure a topic for Version 5 default messaging

Configuring a connection for V5 default messaging

Use this task to configure a connection to V5 default messaging. This task is provided to help you manually migrate nodes from Version 5 to Version 6. If you use the supplied tools to migrate existing Version 5 nodes to Version 6, the jmsserver, application server and port that you create with this task are defined automatically.

About this task

To configure a connection for V5 default messaging, use the administrative console to complete the following steps:

Procedure

1. Create an application server with the name jmsserver (only one of these can exist on each node). In the navigation pane, expand **Servers** > **Server Types** > **WebSphere application servers**.
2. On the new server, define a new JMSSERVER_QUEUED_ADDRESS port with the host set to the Version 6 server host, and the port set to the same as the SIB_MQ_ENDPOINT_ADDRESS of the server that is a member of your bus. The appserver called jmsserver does not have to be started; it is only used for looking up the port address.
3. Define a queue connection factory and queue at the Node or Cell scope. In the navigation pane, click **Resources** > **JMS->JMS providers**.

4. In the content pane, click the name of the V5 default messaging provider.
5. Define a queue destination on your bus with the same name as your queue defined for the V5 default messaging provider.
6. Define an alias destination on your bus with the same name as your queue defined for the V5 default messaging provider but with WQ_ appended to the front the name. For example, if your queue has the name MyV5Queue, your alias should have the name WQ_MyV5Queue.
7. Point the alias at your queue destination with the correct name. The migration process targets the queue with the WQ_ prefix; defining the alias to point to the real queue helps migration.
8. Define the WebSphere MQ Client Link on your messaging engine on the bus. Keep the WebSphere MQ channel name WAS.JMS.SVRCONN and set the queue manager name so that it contains your node name. For example, if your node name is MyNode, you would set the queue manager name to WAS_<MyNode>_jmsserver. Now set the queue manager name to the same; in this example it would be WAS_MyNode_jmsserver. The WebSphere MQ Client Link will show a status of inactive, this is normal.
9. Restart your server so that the new JNDI definitions bind correctly. Your Version 5 client should now be able to connect to Version 6 by using the host and bootstrap address of the Version 6 system in the provider URL component of the initial context. Your client should now also be able to send messages to the destination on the bus.
10. If you open the Client connections view and click the refresh icon, the host name of the connecting system is visible when the client is connected. In the navigation pane, expand**Buses** > **[bus name]** > **Messaging engines** > **[messaging engine name]** > **WebSphere MQ client links** > **[client link name]** > **Runtime** > **Client connections view**.
11. Click **OK**.
12. Save any changes to the master configuration.
13. To have the changed configuration take effect, stop then restart the application server.

Configuring a queue connection factory for Version 5 default messaging

Use this task to browse or change the properties of a JMS queue connection factory for point-to-point messaging with the default messaging provider on a Version 5 node. This task contains an optional step for you to create a new JMS queue connection factory.

About this task

To configure a JMS queue connection factory for use by WebSphere Application Server Version 5 applications, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. Select the V5 default messaging provider for which you want to configure a queue connection factory.
3. In the content pane, under Additional Properties, click **Queue connection factories** This displays any existing JMS queue connection factories for the Version 5 messaging provider in the content pane.
4. To browse or change an existing JMS queue connection factory, click its name in the list. Otherwise, to create a new connection factory, complete the following steps:
   a. Click **New** in the content pane.
   b. Specify the following required properties. You can specify other properties, as described in a later step.

      Name The name by which this JMS queue connection factory is known for administrative purposes within WebSphere Application Server.

      JNDI Name
      The JNDI name that is used to bind the JMS queue connection factory into the namespace.

c. Click **Apply**. This defines the JMS queue connection factory to WebSphere Application Server, and enables you to browse or change additional properties.

5. Optional: Change properties for the queue connection factory, according to your needs.
6. Click **OK**.
7. Save any changes to the master configuration.
8. To have the changed configuration take effect, stop then restart the application server.

Version 5 default messaging queue connection factory settings:

Use this panel to browse or change the configuration properties of the selected JMS queue connection factory for point-to-point messaging for use by WebSphere Application Server Version 5 applications.

A WebSphere queue connection factory is used to create JMS connections to the default messaging provider for use by WebSphere Application Server Version 5 applications.

To view this page, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. **(Optional)** In the content pane, change the **Scope** setting to the level at which JMS resources are visible to applications. If you define a Version 5 JMS resource at the Cell scope level, all users in the cell can look up and use that JMS resource.
3. In the content pane, click the name of the V5 default messaging provider that you want to support the JMS destination.
4. Under Additional Resources, click **Queue connection factories**. This displays a list of any existing JMS queue connection factories.
5. Click the name of the JMS queue connection factory that you want to work with.

A queue connection factory for the embedded WebSphere JMS provider has the following properties:

Scope:

Specifies the level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the messaging provider settings panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Name:

The name by which this JMS queue connection factory is known for administrative purposes within IBM WebSphere Application Server. The name must be unique within the JMS connection factories across the WebSphere administrative domain.

Data type String
Default Null

JNDI name:

The JNDI name that is used to bind the JMS connection factory into the namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Description:

A description of this connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String
Default Null

Category:

A category used to classify or group this connection factory, for your IBM WebSphere Application Server administrative records.

Data type String

Node:

The WebSphere node name of the administrative node where the JMS server runs for this connection factory. Connections created by this factory connect to that JMS server.

Data type Enum
Default Null
Range Drop-down list of Version 5 nodes in the WebSphere administrative domain.

Component-managed Authentication Alias:

This alias specifies a user ID and password to be used to authenticate connection to the messaging provider for application-managed authentication.

This property provides a list of the J2C authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate the creation of a new connection to the JMS provider.

If you have enabled administrative security for WebSphere Application Server, select the alias that specifies the user ID and password used to authenticate the creation of a new connection to the messaging provider. The use of this alias depends on the resource authentication (res-auth) setting declared in the connection factory resource reference of the deployment descriptors for an application component.

Note: User IDs longer than 12 characters cannot be used for authentication with the V5 default messaging provider. For example, the default Windows NT user ID, **Administrator**, is not valid because it contains 13 characters. Therefore, an authentication alias for a WebSphere queue connection factory must specify a user ID no longer than 12 characters.

Container-managed Authentication Alias:

This alias specifies a user ID and password to be used to authenticate connection to the messaging provider for container-managed authentication.

The specification of a login configuration and associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is Container. If the 'DefaultPrincipalMapping' login configuration is used, the associated property is a container-managed authentication alias. This field is used only in the absence of a loginConfiguration on the component resource reference. To define a new alias, see the related item J2EE Connector Architecture (J2C) authentication data entries.

This property provides a list of the J2C authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate the creation of a new connection to the JMS provider.

If you have enabled administrative security for WebSphere Application Server, select the alias that specifies the user ID and password used to authenticate the creation of a new connection to the messaging provider. The use of this alias depends on the resource authentication (res-auth) setting declared in the connection factory resource reference of the deployment descriptors for an application component.

Note: User IDs longer than 12 characters cannot be used for authentication with the V5 default messaging provider. For example, the default Windows NT user ID, **Administrator**, is not valid because it contains 13 characters. Therefore, an authentication alias for a WebSphere topic connection factory must specify a user ID no longer than 12 characters.

Mapping-Configuration Alias:

The module used to map authentication aliases.

This field is deprecated in 6.0. The specification of a login configuration and associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is Container. This field is used only in the absence of a loginConfiguration on the component resource reference.

This field provides a list of the modules that have been configured on the **Security** > **JAAS Configuration** > **Application Logins Configuration** property. For more information about the mapping configurations, see Java Authentication and Authorization service configuration entry settings.

Data type Enum

Default Null

Range

ClientContainer
The client container maps authentication aliases.

WSLogin
The WSLogin module maps authentication aliases.

DefaultPrincipalMapping
The JAAS configuration maps an authentication alias to its userid and password.

XA Enabled:

Specifies whether the connection factory is for XA or non-XA coordination of messages and controls if the application server uses XA QCF/TCF. Enable XA if multiple resources are used in the same transaction.

If you clear this check box property (for non-XA coordination), the JMS session is still enlisted in a transaction, but uses the resource manager local transaction calls (session.commit and session.rollback) instead of XA calls. This can lead to an improvement in performance. However, this means that only a single resource can be enlisted in a transaction in WebSphere Application Server.

Last participant support enables you to enlist one non-XA resource with other XA-capable resources.

For a WebSphere Topic Connection Factory with the **Port** property set to DIRECT this property does not apply, and always adopts non-XA coordination.

Data type	check box
Default	Selected (enabled for XA coordination)
Range	**Selected** The connection factory is enabled for XA-coordination of messages **Cleared** The connection factory is not enabled for XA coordination of messages
Recommended	Do not enable XA coordination when the message queue or topic received is the only resource in the transaction. Enable XA coordination when other resources, including other queues or topics, are involved.

Connection pool:

Specifies an optional set of connection pool settings.

Connection pool properties are common to all J2C connectors.

The application server pools connections and sessions with the messaging provider to improve performance. You have to configure the connection and session pool properties appropriately for your applications, otherwise you may not get the connection and session behavior that you want.

Change the size of the connection pool if concurrent server-side access to the JMS resource exceeds the default value. The size of the connection pool is set on a per queue or topic basis.

Session pool:

An optional set of session pool settings.

This link provides a panel of optional connection pool properties, common to all J2C connectors.

The application server pools connections and sessions with the messaging provider to improve performance. You have to configure the connection and session pool properties appropriately for your applications, otherwise you may not get the connection and session behavior that you want.

Session pool settings:

Use this page to configure session pool settings.

This administrative console page is common to JMS unified connection factories, queue connection factories and topic connection factories. To view this page, you select an instance of the resource type then click **Session pools**. For example, click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Session pools**.

Connection Timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

The wait is necessary when the maximum value of connections (**Max Connections**) to a particular connection pool is reached . For example, if *Connection Timeout* is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is *not* available within this time, the Pool Manager throws a ConnectionWaitTimeoutException. It usually does not make sense to retry the getConnection() method, because if a longer wait time is required, you should set the **Connection Timeout** setting to a higher value. Therefore, if this exception is caught by the application, the administrator should review the expected usage of the application and tune the connection pool and the database accordingly.

If Connection Timeout is set to 0, the Pool Manager waits as long as necessary until a connection is allocated (which happens when the number of connections falls below the value of **Max Connections**).

If Max Connections is set to 0, which enables an infinite number of physical connections, then the Connection Timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Max Connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use returns to the pool, or a ConnectionWaitTimeoutException is thrown.

For example, if the Max Connections value is set to 5, and there are five physical connections in use, the pool manager waits for the amount of time specified in Connection Timeout for a physical connection to become free.

If Max Connections is set to 0, the Connection Timeout value is ignored.

For better performance, set the value for the connection pool lower than the value for the Max Connections option in the web container. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

If clones are used, one data pool exists for each clone. Knowing the number of data pools is important when configuring the database maximum connections.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the CPU load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Data type	Integer
Default	10
Range	0 to max int

Min Connections:

Specifies the minimum number of physical connections to maintain.

Until this number is reached, the pool maintenance thread does not discard physical connections. However, no attempt is made to bring the number of connections up to this number. If you set a value for Aged Timeout, the minimum is not maintained. All connections with an expired age are discarded.

For example if the **Min Connections** value is set to 3, and one physical connection is created, the Unused Timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the **Min Connections** setting.

Data type	Integer
Default	1
Range	0 to max int

Reap Time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if **Reap Time** is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in **Min Connections**. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread set Reap Time to 0, or set both Unused Timeout and Aged Timeout to 0. The recommended way to disable the pool maintenance thread is to set Reap Time to 0, in which case Unused Timeout and Aged Timeout are ignored. However, if Unused Timeout and Aged Timeout are set to 0, the pool maintenance thread runs, but only physical connections that timeout due to non-zero timeout values are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused Timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the **Min Connections** setting. For example, if the unused timeout value is set to 120, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the **Reap Time** value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged Timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting **Aged Timeout** to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged Timeout value higher than the **Reap Timeout** value for optimal performance. For example, if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, are affected by the Reap Time value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge Policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**. Java EE Connector Architecture (JCA) data sources can have either option. WebSphere Version 4.0 data sources always have a purge policy of **EntirePool**.

Data type	String
Default	FailingConnectionOnly
Range	**EntirePool** All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and throws a *StaleConnectionException* during the next operation on that connection. Subsequent *getConnection* requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice. **FailingConnectionOnly** Only the connection that caused the *StaleConnectionException* is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a good possibility that the next *getConnection* request from the application can return a connection from the pool that is also stale, resulting in more stale connection exceptions.

Configuring a topic connection factory for Version 5 default messaging

Use this task to browse or change a JMS topic connection factory for publish/subscribe messaging by WebSphere Application Server Version 5 applications.

About this task

To configure a JMS topic connection factory for use by WebSphere Application Server Version 5 applications, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. Select the V5 default messaging provider for which you want to configure a topic connection factory.
3. Optional: Change the **Scope** setting to the level at which the JMS topic connection factory is visible to applications. If you define a Version 5 JMS resource at the Cell scope level, all users in the cell can look up and use that JMS resource.
4. In the content pane, under Additional Properties, click **Topic connection factories** This displays any existing JMS topic connection factories for the Version 5 messaging provider in the content pane.
5. To browse or change an existing JMS topic connection factory, click its name in the list. Otherwise, to create a new connection factory, complete the following steps:
   a. Click **New** in the content pane.
   b. Specify the following required properties. You can specify other properties, as described in a later step.

      Name The name by which this JMS topic connection factory is known for administrative purposes within IBM WebSphere Application Server.

      JNDI Name
      The JNDI name that is used to bind the JMS topic connection factory into the namespace.
   c. Click **Apply**. This defines the JMS topic connection factory to WebSphere Application Server, and enables you to browse or change additional properties.
6. Optional: Change properties for the topic connection factory, according to your needs.
7. Click **OK**.
8. Save any changes to the master configuration.
9. To have the changed configuration take effect, stop then restart the application server.

Version 5 default messaging topic connection factory settings:

Use this panel to browse or change the configuration properties of the selected JMS topic connection factory for publish/subscribe messaging by WebSphere Application Server Version 5 applications.

A WebSphere topic connection factory is used to create JMS connections to the default messaging provider for use by WebSphere Application Server Version 5 applications.

To view this page, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. **(Optional)** In the content pane, change the **Scope** setting to the level at which JMS resources are visible to applications. If you define a Version 5 JMS resource at the Cell scope level, all users in the cell can look up and use that JMS resource.
3. In the content pane, click the name of the V5 default messaging provider that you want to support the JMS destination.
4. Under Additional Resources, click **Topic connection factories**. This displays a list of any existing JMS topic connection factories.
5. Click the name of the JMS topic connection factory that you want to work with.

A JMS topic connection factory for use with the V5 default messaging provider has the following properties.

Scope:

Specifies the level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the messaging provider settings panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Name:

The name by which this JMS topic connection factory is known for administrative purposes within IBM WebSphere Application Server. The name must be unique within the JMS connection factories across the WebSphere administrative domain.

Data type String
Default Null

JNDI name:

The JNDI name that is used to bind the topic connection factory into the namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Description:

A description of this topic connection factory for administrative purposes within IBM WebSphere Application Server.

Data type String
Default Null

Category:

A category used to classify or group this topic connection factory, for your IBM WebSphere Application Server administrative records.

Data type String

Node:

The WebSphere node name of the administrative node where the JMS server runs for this connection factory. Connections created by this factory connect to that JMS server.

Data type Enum
Default Null
Range Drop-down list of nodes in the WebSphere administrative domain.

Port:

Which of the two ports that connections use to connect to the JMS server. The QUEUED port is for full-function JMS publish/subscribe support, the DIRECT port is for non-persistent, non-transactional, non-durable subscriptions only.

Note: Message-driven beans cannot use the direct listener port for publish/subscribe support. Therefore, any topic connection factory configured with **Port** set to Direct cannot be used with message-driven beans.

Data type Enum
Units Not applicable
Default QUEUED
Range
QUEUED
The listener port used for full-function JMS-compliant, publish/subscribe support.
DIRECT
The listener port used for direct TCP/IP connection (non-transactional, non-persistent, and non-durable subscriptions only) for publish/subscribe support.

Component-managed Authentication Alias:

This alias specifies a user ID and password to be used to authenticate connection to the messaging provider for application-managed authentication.

This property provides a list of the J2C authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate the creation of a new connection to the JMS provider.

If you have enabled administrative security for WebSphere Application Server, select the alias that specifies the user ID and password used to authenticate the creation of a new connection to the messaging provider. The use of this alias depends on the resource authentication (res-auth) setting declared in the connection factory resource reference of the deployment descriptors for an application component.

Note: User IDs longer than 12 characters cannot be used for authentication with the V5 default messaging provider. For example, the default Windows NT user ID, **Administrator**, is not valid because it contains 13 characters. Therefore, an authentication alias for a WebSphere topic connection factory must specify a user ID no longer than 12 characters.

Container-managed Authentication Alias:

This alias specifies a user ID and password to be used to authenticate connection to the messaging provider for container-managed authentication.

The specification of a login configuration and associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is Container. If the 'DefaultPrincipalMapping' login configuration is used, the associated property is a container-managed authentication alias. This field is used only in the absence of a loginConfiguration on the component resource reference. To define a new alias, see the related item J2EE Connector Architecture (J2C) authentication data entries.

This property provides a list of the J2C authentication data entry aliases that have been defined to WebSphere Application Server. You can select a data entry alias to be used to authenticate the creation of a new connection to the JMS provider.

If you have enabled administrative security for WebSphere Application Server, select the alias that specifies the user ID and password used to authenticate the creation of a new connection to the JMS provider. The use of this alias depends on the resource authentication (res-auth) setting declared in the connection factory resource reference of the deployment descriptors for an application component.

Note: User IDs longer than 12 characters cannot be used for authentication with the embedded WebSphere JMS provider. For example, the default Windows NT user ID, **Administrator**, is not valid for use with embedded WebSphere messaging, because it contains 13 characters. Therefore, an authentication alias for a WebSphere JMS provider connection factory must specify a user ID no longer than 12 characters.

Mapping-Configuration Alias:

The module used to map authentication aliases.

The specification of a login configuration and associated properties on the component resource reference determines the container-managed authentication strategy when the res-auth value is Container. This field is used only in the absence of a loginConfiguration on the component resource reference.

This field provides a list of the modules that have been configured on the **Security** > **JAAS Configuration** > **Application Logins Configuration** property. For more information about the mapping configurations, see Java Authentication and Authorization service configuration entry settings.

Data type Enum
Default Null
Range
ClientContainer
The client container maps authentication aliases.
WSLogin
The WSLogin module maps authentication aliases.
DefaultPrincipalMapping
The JAAS configuration maps an authentication alias to its userid and password.

Clone Support:

Select this check box to enable clone support to allow the same durable subscription across topic clones.

Data type Enum
Default Cleared
Range
Selected
Clone support is enabled.
Cleared
Clone support is disabled.

If you select this property, you must also specify a value for the **Client ID** property.

Client ID:

The JMS client identifier used for connections to the queue manager.

Data type	String
Range	A valid JMS client ID

XA Enabled:

Specifies whether the connection factory is for XA or non-XA coordination of messages and controls if the application server uses XA QCF/TCF. Enable XA if multiple resources are used in the same transaction.

If you clear this check box property (for non-XA coordination), the JMS session is still enlisted in a transaction, but uses the resource manager local transaction calls (session.commit and session.rollback) instead of XA calls. This can lead to an improvement in performance. However, this means that only a single resource can be enlisted in a transaction in WebSphere Application Server.

Last participant support enables you to enlist one non-XA resource with other XA-capable resources.

For a WebSphere Topic Connection Factory with the **Port** property set to DIRECT this property does not apply, and always adopts non-XA coordination.

Data type	check box
Default	Selected (enabled for XA coordination)
Range	**Selected** The connection factory is enabled for XA-coordination of messages **Cleared** The connection factory is not enabled for XA coordination of messages
Recommended	Do not enable XA coordination when the message queue or topic received is the only resource in the transaction. Enable XA coordination when other resources, including other queues or topics, are involved.

Connection pool:

Specifies an optional set of connection pool settings.

Connection pool properties are common to all J2C connectors.

The application server pools connections and sessions with the messaging provider to improve performance. You have to configure the connection and session pool properties appropriately for your applications, otherwise you might not get the connection and session behavior that you want.

Change the size of the connection pool if concurrent server-side access to the JMS resource exceeds the default value. The size of the connection pool is set on a per queue or topic basis.

Session pool:

An optional set of session pool settings.

This link provides a panel of optional connection pool properties, common to all J2C connectors.

The application server pools connections and sessions with the JMS provider to improve performance. You have to configure the connection and session pool properties appropriately for your applications, otherwise you might not get the connection and session behavior that you want.

Session pool settings:

Use this page to configure session pool settings.

This administrative console page is common to JMS unified connection factories, queue connection factories and topic connection factories. To view this page, you select an instance of the resource type then click **Session pools**. For example, click **Resources** > **JMS->Queue connection factories->*queue_connection_factory*->[Additional Properties] Session pools**.

Connection Timeout:

Specifies the interval, in seconds, after which a connection request times out and a ConnectionWaitTimeoutException is thrown.

The wait is necessary when the maximum value of connections (**Max Connections**) to a particular connection pool is reached . For example, if *Connection Timeout* is set to 300 and the maximum number of connections is reached, the Pool Manager waits for 300 seconds for an available physical connection. If a physical connection is *not* available within this time, the Pool Manager throws a ConnectionWaitTimeoutException. It usually does not make sense to retry the getConnection() method, because if a longer wait time is required, you should set the **Connection Timeout** setting to a higher value. Therefore, if this exception is caught by the application, the administrator should review the expected usage of the application and tune the connection pool and the database accordingly.

If Connection Timeout is set to 0, the Pool Manager waits as long as necessary until a connection is allocated (which happens when the number of connections falls below the value of **Max Connections**).

If Max Connections is set to 0, which enables an infinite number of physical connections, then the Connection Timeout value is ignored.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Max Connections:

Specifies the maximum number of physical connections that you can create in this pool.

These are the physical connections to the backend resource. Once this number is reached, no new physical connections are created and the requester waits until a physical connection that is currently in use returns to the pool, or a ConnectionWaitTimeoutException is thrown.

For example, if the Max Connections value is set to 5, and there are five physical connections in use, the pool manager waits for the amount of time specified in Connection Timeout for a physical connection to become free.

If Max Connections is set to 0, the Connection Timeout value is ignored.

For better performance, set the value for the connection pool lower than the value for the Max Connections option in the web container. Lower settings, such as 10-30 connections, perform better than higher settings, such as 100.

If clones are used, one data pool exists for each clone. Knowing the number of data pools is important when configuring the database maximum connections.

You can use the Tivoli Performance Viewer to find the optimal number of connections in a pool. If the number of concurrent waiters is greater than 0, but the CPU load is not close to 100%, consider increasing the connection pool size. If the Percent Used value is consistently low under normal workload, consider decreasing the number of connections in the pool.

Data type	Integer
Default	10
Range	0 to max int

Min Connections:

Specifies the minimum number of physical connections to maintain.

Until this number is reached, the pool maintenance thread does not discard physical connections. However, no attempt is made to bring the number of connections up to this number. If you set a value for Aged Timeout, the minimum is not maintained. All connections with an expired age are discarded.

For example if the **Min Connections** value is set to 3, and one physical connection is created, the Unused Timeout thread does not discard that connection. By the same token, the thread does not automatically create two additional physical connections to reach the **Min Connections** setting.

Data type	Integer
Default	1
Range	0 to max int

Reap Time:

Specifies the interval, in seconds, between runs of the pool maintenance thread.

For example, if **Reap Time** is set to 60, the pool maintenance thread runs every 60 seconds. The Reap Time interval affects the accuracy of the **Unused Timeout** and **Aged Timeout** settings. The smaller the interval, the greater the accuracy. If the pool maintenance thread is enabled, set the Reap Time value less than the values of Unused Timeout and Aged Timeout. When the pool maintenance thread runs, it discards any connections remaining unused for longer than the time value specified in Unused Timeout, until it reaches the number of connections specified in **Min Connections**. The pool maintenance thread also discards any connections that remain active longer than the time value specified in Aged Timeout.

The Reap Time interval also affects performance. Smaller intervals mean that the pool maintenance thread runs more often and degrades performance.

To disable the pool maintenance thread set Reap Time to 0, or set both Unused Timeout and Aged Timeout to 0. The recommended way to disable the pool maintenance thread is to set Reap Time to 0, in which case Unused Timeout and Aged Timeout are ignored. However, if Unused Timeout and Aged Timeout are set to 0, the pool maintenance thread runs, but only physical connections that timeout due to non-zero timeout values are discarded.

Data type	Integer
Units	Seconds
Default	180
Range	0 to max int

Unused Timeout:

Specifies the interval in seconds after which an unused or idle connection is discarded.

Set the Unused Timeout value higher than the Reap Timeout value for optimal performance. Unused physical connections are only discarded if the current number of connections not in use exceeds the **Min Connections** setting. For example, if the unused timeout value is set to 120, and the pool maintenance thread is enabled (Reap Time is not 0), any physical connection that remains unused for two minutes is discarded. Note that accuracy of this timeout, as well as performance, is affected by the **Reap Time** value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	1800
Range	0 to max int

Aged Timeout:

Specifies the interval in seconds before a physical connection is discarded.

Setting **Aged Timeout** to 0 supports active physical connections remaining in the pool indefinitely. Set the Aged Timeout value higher than the **Reap Timeout** value for optimal performance. For example, if the Aged Timeout value is set to 1200, and the Reap Time value is not 0, any physical connection that remains in existence for 1200 seconds (20 minutes) is discarded from the pool. Note that accuracy of this timeout, as well as performance, are affected by the Reap Time value. For more information, see Reap Time.

Data type	Integer
Units	Seconds
Default	0
Range	0 to max int

Purge Policy:

Specifies how to purge connections when a *stale connection* or *fatal connection error* is detected.

Valid values are **EntirePool** and **FailingConnectionOnly**. Java EE Connector Architecture (JCA) data sources can have either option. WebSphere Version 4.0 data sources always have a purge policy of **EntirePool**.

Data type	String
Default	FailingConnectionOnly

Range

EntirePool
All connections in the pool are marked stale. Any connection not in use is immediately closed. A connection in use is closed and throws a *StaleConnectionException* during the next operation on that connection. Subsequent *getConnection* requests from the application result in new connections to the database opening. When using this purge policy, there is a slight possibility that some connections in the pool are closed unnecessarily when they are not stale. However, this is a rare occurrence. In most cases, a purge policy of EntirePool is the best choice.

FailingConnectionOnly
Only the connection that caused the *StaleConnectionException* is closed. Although this setting eliminates the possibility that valid connections are closed unnecessarily, it makes recovery from an application perspective more complicated. Because only the currently failing connection is closed, there is a good possibility that the next *getConnection* request from the application can return a connection from the pool that is also stale, resulting in more stale connection exceptions.

Configuring a queue for Version 5 default messaging

Use this task to browse or change the properties of a JMS queue destination for point-to-point messaging by WebSphere Application Server Version 5 applications. This task contains an optional step for you to create a new topic destination.

About this task

To configure a JMS queue destination for use by WebSphere Application Server Version 5 applications, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. Select the V5 default messaging provider for which you want to configure a queue destination.
3. Optional: Change the **Scope** setting to the level at which the JMS destination is visible to applications.
4. In the content pane, under Additional Properties, click **Queues** This displays any existing queue destinations for the Version 5 default messaging provider in the content pane.
5. To browse or change an existing JMS queue destination, click its name in the list. Otherwise, to create a new queue destination, complete the following steps:
   a. Click **New** in the content pane.
   b. Specify the following required properties. You can specify other properties, as described in a later step.

      Name The name by which this queue destination is known for administrative purposes within IBM WebSphere Application Server.

      JNDI Name
      The JNDI name that is used to bind the queue destination into the namespace.

c. Click **Apply**. This defines the queue destination to WebSphere Application Server, and enables you to browse or change additional properties.

6. Optional: Change properties for the queue destination, according to your needs.
7. Click **OK**.
8. Save any changes to the master configuration.
9. To make a queue destination available to applications, host the queue on a JMS server. To add a new queue to a JMS server or to change an existing queue on a JMS server, you define the administrative name of the queue to the JMS server.
10. To have the changed configuration take effect, stop then restart the application server.

Version 5 default messaging queue settings:

Use this page to view or change the configuration properties of the selected JMS queue destination for point-to-point messaging by WebSphere Application Server Version 5 applications.

A queue destination is used to configure a JMS queue of the default messaging provider for use by WebSphere Application Server Version 5 applications. Connections to the queue are created by the associated V5 default messaging WebSphere queue connection factory.

To view this page, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. **(Optional)** In the content pane, change the **Scope** setting to the level at which JMS resources are visible to applications. If you define a Version 5 JMS resource at the Cell scope level, all users in the cell can look up and use that JMS resource. However, the JMS resource has only the subset of properties that apply to WebSphere Application Server Version 5. If you want to define a JMS resource at Cell level for use on non-Version 5 nodes, you should define the JMS resource for the Version 6 default messaging provider.
3. In the content pane, click the name of the V5 default messaging provider that you want to support the JMS destination.
4. Under Additional Resources, click **Queues**. This displays a list of any existing JMS queue destinations.
5. Click the name of the JMS queue destination that you want to work with.

A JMS queue for use with the internal WebSphere JMS provider has the following properties.

Scope:

Specifies the level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this page. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the JMS provider settings page, then click **Apply**, before clicking the link for the type of resource.

Data type String

Name:

The name by which the queue is known for administrative purposes within WebSphere Application Server.

To enable applications to use this queue, you must add the queue name to the **Queue Names** field on the Version 5 JMS server settings page for the JMS server that hosts the queue.

Data type String

JNDI name:

The JNDI name that is used to bind the queue into the application server namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Description:

A description of the queue, for administrative purposes.

Data type String
Default Null

Category:

A category used to classify or group this queue, for your WebSphere Application Server administrative records.

Data type String

Persistence:

Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application.

Data type Enum
Default APPLICATION DEFINED
Range

APPLICATION DEFINED
Messages on the destination have their persistence defined by the application that put them onto the queue.

NON PERSISTENT
Messages on the destination are not persistent.

PERSISTENT
Messages on the destination are persistent. When a persistent message is put to a queue, all of the message data is written to the messaging log (under the *embedded_messaging_install*\log directory) to make recovery of the message possible.

Priority:

Whether the message priority for this destination is defined by the application or the **Specified priority** property

Data type Enum
Default APPLICATION DEFINED
Range
APPLICATION DEFINED
The priority of messages on this destination is defined by the application that put them onto the destination.
QUEUE DEFINED
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.
SPECIFIED
The priority of messages on this destination is defined by the **Specified priority** property.
Note: If you select this option, you must define a priority on the **Specified priority** property.

Specified priority:

If the **Priority** property is set to Specified, type here the message priority for this queue, in the range 0 (lowest) through 9 (highest)

If the **Priority** property is set to Specified, messages sent to this queue have the priority value specified by this property.

Data type Integer
Units Message priority level
Default 0
Range 0 (lowest priority) through 9 (highest priority)

Expiry:

Whether the expiry timeout for this queue is defined by the application or the **Specified expiry** property, or messages on the queue never expire (have an unlimited expiry timeout)

Data type Enum
Default APPLICATION DEFINED
Range
APPLICATION DEFINED
The expiry timeout for messages on this queue is defined by the application that put them onto the queue.
UNLIMITED
Messages on this queue have no expiry timeout, so those messages never expire.
SPECIFIED
The expiry timeout for messages on this queue is defined by the **Specified expiry** property.
Note: If you select this option, you must define a timeout on the **Specified expiry** property.

Specified expiry:

If the **Expiry timeout** property is set to Specified, type here the number of milliseconds (greater than 0) after which messages on this queue expire

Data type Integer
Units Milliseconds

Default 0

Range Greater than or equal to 0
- 0 indicates that messages never timeout
- Other values are an integer number of milliseconds

Configuring a topic for Version 5 default messaging

Use this task to browse or change the properties of a JMS topic destination for publish/subscribe messaging by WebSphere Application Server Version 5 applications. This task contains an optional step for you to create a new topic destination.

About this task

To configure a JMS topic destination for use WebSphere Application Server Version 5 applications, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Resources** > **JMS->JMS providers**.
2. Select the V5 default messaging provider for which you want to configure a topic destination.
3. Optional: Change the **Scope** setting to the level at which the JMS destination is visible to applications. If you define a Version 5 JMS resource at the Cell scope level, all users in the cell can look up and use that JMS resource.
4. In the content pane, under Additional Properties, click **Topics** This displays any existing JMS topic destinations for the Version 5 default messaging provider in the content pane.
5. To browse or change an existing JMS topic destination, click its name in the list. Otherwise, to create a new topic destination, complete the following steps:
   a. Click **New** in the content pane.
   b. Specify the following required properties. You can specify other properties, as described in a later step.

      Name The name by which this topic destination is known for administrative purposes within IBM WebSphere Application Server.

      JNDI Name
      The JNDI name that is used to bind the topic destination into the namespace.

      Topic The name of the topic in the default messaging provider, to which messages are sent.
   c. Click **Apply**. This defines the topic destination to WebSphere Application Server, and enables you to browse or change additional properties.
6. Optional: Change properties for the topic destination, according to your needs.
7. Click **OK**.
8. Save any changes to the master configuration.
9. To have the changed configuration take effect, stop then restart the application server.

Version 5 default messaging topic settings:

Use this panel to browse or change the configuration properties of the selected JMS topic destination for publish/subscribe messaging by WebSphere Application Server Version 5 applications.

A WebSphere topic destination is used to configure the properties of a JMS topic for the default messaging provider on a Version 5 node. Connections to the topic are created by the associated topic connection factory.

To view this page, use the administrative console to complete the following steps:
1. In the navigation pane, click **Resources** > **JMS->JMS providers**.

2. **(Optional)** In the content pane, change the **Scope** setting to the level at which JMS resources are visible to applications. If you define a Version 5 JMS resource at the Cell scope level, all users in the cell can look up and use that JMS resource. However, the JMS resource has only the subset of properties that apply to WebSphere Application Server Version 5. If you want to define a JMS resource at Cell level for use on non-Version 5 nodes, you should define the JMS resource for the Version 6 default messaging provider.
3. In the content pane, click the name of the V5 default messaging provider that you want to support the JMS destination.
4. Under Additional Resources, click **Topics**. This displays a list of any existing JMS topic destinations.
5. Click the name of the JMS topic destination that you want to work with.

A JMS topic destination for use with the V5 default messaging provider has the following properties.

Scope:

Specifies the level to which this resource definition is visible to applications.

Resources such as messaging providers, namespace bindings, or shared libraries can be defined at multiple scopes, with resources defined at more specific scopes overriding duplicates that are defined at more general scopes.

The scope displayed is for information only, and cannot be changed on this panel. If you want to browse or change this resource (or other resources) at a different scope, change the scope on the messaging provider settings panel, then click **Apply**, before clicking the link for the type of resource.

Data type String

Name:

The name by which the topic is known for administrative purposes within IBM WebSphere Application Server.

Data type String

JNDI name:

The JNDI name that is used to bind the topic into the application server namespace.

As a convention, use the fully qualified JNDI name; for example, in the form jms/*Name*, where *Name* is the logical name of the resource.

This name is used to link the platform binding information. The binding associates the resources defined by the deployment descriptor of the module to the actual (physical) resources bound into JNDI by the platform.

Data type String

Description:

A description of the topic, for administrative purposes within IBM WebSphere Application Server.

Data type String
Default Null

Category:

A category used to classify or group this topic, for your IBM WebSphere Application Server administrative records.

Data type String

Topic:

The name of the topic as defined to the JMS provider.

Data type String
Default Null
Range The topic value can be dot notation and include wildcard characters.

Persistence:

Whether all messages sent to the destination are persistent, non-persistent, or have their persistence defined by the application

Data type Enum
Default APPLICATION DEFINED
Range
APPLICATION DEFINED
Messages on the destination have their persistence defined by the application that put them onto the queue.
NON-PERSISTENT
Messages on the destination are not persistent.
PERSISTENT
Messages on the destination are persistent.

Priority:

Whether the message priority for this destination is defined by the application or the **Specified priority** property

Data type Enum
Units Not applicable
Default APPLICATION DEFINED
Range
APPLICATION DEFINED
The priority of messages on this destination is defined by the application that put them onto the destination.
QUEUE DEFINED
[WebSphere MQ destination only] Messages on the destination have their persistence defined by the WebSphere MQ queue definition properties.
SPECIFIED
The priority of messages on this destination is defined by the **Specified priority** property. *If you select this option, you must define a priority on the **Specified priority** property.*

Specified priority:

If the **Priority** property is set to Specified, type here the message priority for this queue, in the range 0 (lowest) through 9 (highest)

If the **Priority** property is set to Specified, messages sent to this queue have the priority value specified by this property.

Data type Integer
Units Message priority level
Default 0
Range 0 (lowest priority) through 9 (highest priority)

Expiry:

Whether the expiry timeout for this queue is defined by the application or the **Specified expiry** property, or messages on the queue never expire (have an unlimited expiry timeout)

Data type Enum
Units Not applicable
Default APPLICATION DEFINED
Range

APPLICATION DEFINED
The expiry timeout for messages on this queue is defined by the application that put them onto the queue.

UNLIMITED
Messages on this queue have no expiry timeout, so those messages never expire.

SPECIFIED
The expiry timeout for messages on this queue is defined by the **Specified expiry** property. *If you select this option, you must define a timeout on the **Specified expiry** property.*

Specified expiry:

If the **Expiry timeout** property is set to Specified, type here the number of milliseconds (greater than 0) after which messages on this queue expire

Data type Integer
Units Milliseconds
Default 0
Range Greater than or equal to 0

- 0 indicates that messages never timeout
- Other values are an integer number of milliseconds

Chapter 15. Managing message-driven beans

You can manage the Java EE Connector Architecture (JCA) Version 1.5-compliant message-driven beans that you deploy as message endpoints, and you can manage the message listener resources for non-JCA message-driven beans that you deploy against listener ports.

Before you begin

For WebSphere Application Server Version 7 and later, listener ports are stabilized. For more information, read the article on stabilized features. You should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications. However, you should not begin this migration until you are sure the application does not have to work on application servers earlier than WebSphere Application Server Version 7. For example, if you have an application server cluster with some members at Version 6.1 and some at Version 7, you should not migrate applications on that cluster to use activation specifications until after you migrate all the application servers in the cluster to Version 7.

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

You can manage the following resources for message-driven beans:

- JCA 1.5-compliant message-driven beans that you deploy as message endpoints, and the associated activation specifications.
- The message listener service, listener ports, and listeners for non-JCA message-driven beans that you deploy against listener ports.

Procedure

- Manage JCA 1.5-compliant message-driven beans that are used as message endpoints.

  JCA 1.5-compliant message-driven beans, deployed by using activation specifications, can be used as message endpoints. You can start and stop specific endpoints within your applications to ensure that messages are delivered only to listening message-driven beans that are interacting with healthy resources.
- Manage message listener resources for message-driven beans.

  The message listener service supports message-driven beans that are used with a non-JCA messaging provider. A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. When you deploy a message-driven bean, you associate the bean with a listener port. When a message arrives on the destination, the listener passes the message to a new instance of a message-driven bean for processing. You can manage the resources used by the message listener service, including being able to start and stop specific listener ports manually.

Managing messages with message endpoints

Manage message delivery for message-driven beans (MDB) that are deployed as message endpoints. The message endpoints are managed beans (MBeans) for inbound resource adapters that are compliant with Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) Version 1.5.

About this task

The application server provides message endpoint MBeans to assist you in managing the delivery of a message to your message-driven beans that are acting as listeners on specific endpoints, which are destinations, and in managing the enterprise information system (EIS) resources that are utilized by these message-driven beans. Message-driven beans that are deployed as message endpoints are not the same as message-driven beans that are configured against a listener port. Message-driven beans that are used as message endpoints must be deployed using an ActivationSpecification that is defined within a resource adapter configuration for JCA Version 1.5.

With message endpoint MBeans, you can activate and deactivate specific endpoints within your applications to ensure that messages are delivered only to listening message-driven beans that are interacting with healthy EIS resources. This capability allows you to optimize the performance of your JMS applications in situations where an EIS resource is not behaving as expected. Message delivery to an endpoint typically fails when the message driven bean that is listening invokes an operation against a resource that is not healthy. For example, a messaging provider, which is an inbound resource adapter that is JCA Version 1.5 compliant, might fail to deliver messages to an endpoint when its underlying message-driven bean attempts to commit transactions against a database server that is not responding.

Note: Design your message-driven beans to delegate business processing to other enterprise beans. Do not access the EIS resources directly in the message-driven bean, but do so indirectly through a delegate bean.

Message endpoint MBeans alleviate two problems that are inherent to applications that provide message endpoints that access resources:

- Failed messages require additional processing, such as delivering them to the listening endpoint again or redirecting them to alternate destinations that process failed messages. In addition, a resource adapter might redeliver a message to an endpoint an infinite number of times.
- Message redirection requires the implementation of specialized destinations (queues and listeners) to process failed messages, as well as the logic to detect message failures. Message redirection is potentially error prone and computationally expensive due to its complexity.

The capability to deactivate (pause) and reactivate (resume) a specific message endpoint alleviates these problems by enabling the administrator to deactivate the endpoint from processing messages that are destined to fail. When the message endpoint is deactivated, you can repair the resource that is causing the problems and reactivate the endpoint to resume handling message requests. In the course of troubleshooting, you will not affect the resource adapter or the application that is hosting the endpoint.

If you are connecting to WebSphere MQ, you can also use the WAS_EndpointInitialState custom property in the activation specification to make the message endpoint start out in a deactivated state. When you set this property to Inactive, the message-driven bean connects with the destination, but does not start receiving messages. Use this setting to automatically deactivate a message endpoint when you know that certain tasks must be completed, services must be started, or checks must be carried out, before message handling begins. You activate the message endpoint in the same way as you would reactivate a message endpoint that you paused during its operation.

Procedure

1. Using the administrative console, navigate to the Message Endpoints panel for the application that is hosting the message endpoint.
   a. Select the **Applications** > **Application Types** > **Websphere enterprise applications** > ***application_name***.
   b. Select the **Runtime** panel.
   c. Select **Message Endpoints**. The panel lists the set of message endpoints that are hosted by the application.

2. Optional: Temporarily disable a message endpoint from handling messages and troubleshoot the problem.
   a. Deactivate the message endpoint by selecting the appropriate endpoint and clicking **Pause**.
   b. When the message endpoint is inactive, diagnose and repair the underlying cause of the delivery failures.
   c. Reactivate the message endpoint by selecting the appropriate endpoint and clicking **Resume**.
3. Optional: Activate a message endpoint that started out in a deactivated state. Select the appropriate endpoint and click **Resume**.

Results

The behavior you will observe when you deactivate (pause) a message endpoint using the message endpoint MBean is dependent upon a variety of factors, including the resource adapter that manages the message endpoint, the configuration of the message endpoint and the application server topology. Some specific examples of interest are as follows:

- **MDB listening on a non-durable topic (dependent on configuration):** The behavior that is implied by the deactivation (pause) of a message endpoint is often dependent upon the function that it is fulfilling. For example, if you have configured a message-driven bean to listen on a non-durable topic on the service integration bus, deactivating the message endpoint is analogous to stopping the application and will cause the subscription to be closed. This means that any messages that are published during the time that the message endpoint is paused will not be received by the message-driven bean.
- **Clustered message-driven bean (dependent on topology):** In this scenario a message-driven bean application has been deployed to a cluster of servers. A given message endpoint MBean controls only the behavior of the MDB in one server from the cluster, so will cause only one server to stop processing messages. Depending upon the messaging configuration and the specific resource adapter in use the messages that would have been consumed by the paused message endpoint may be consumed by the active message endpoints in the cluster, or they may remain unconsumed until the paused message endpoint is resumed.
- **Clustered message-driven bean, a non-clustered queue:** In this scenario, you have a cluster of servers with the same message-driven bean deployed to them. This is similar to the case, in which you have different message-driven beans with the same message selection criteria, except that in this case the message-driven beans are logically the same message-driven bean. Pausing the endpoint will cause only one of the servers to stop receiving messages, and the other message-driven beans will receive all the messages; none of the messages will be orphaned. To stop all of the endpoints, you must direct each server in the cluster to stop the local message endpoint.
- **Clustered message-driven bean, clustered queue:** In this scenario, each message-driven bean is pulling messages from a different partition of the queue. Messaging through WebSphere MQ and the Service Integration Bus have similar, but different, capabilities. If you are using WebSphere MQ, then pausing one endpoint will not allow the other instances of the message-driven bean to receive the messages. In the Service Integration Bus, messages from a paused endpoint will be redirected to the other message-driven beans.

Managing message listener resources for message-driven beans

Manage the resources used by the message listener service to support message-driven beans, typically for use with a messaging provider that does not have a Java EE Connector Architecture (JCA) 1.5 resource adapter.

Before you begin

For WebSphere Application Server Version 7 and later, listener ports are stabilized. For more information, read the article on stabilized features. You should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications. However, you should not begin this migration until you are sure the application does not have to work on application

servers earlier than WebSphere Application Server Version 7. For example, if you have an application server cluster with some members at Version 6.1 and some at Version 7, you should not migrate applications on that cluster to use activation specifications until after you migrate all the application servers in the cluster to Version 7.

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

The message listener service is an extension to the JMS functions of the JMS provider and provides a listener manager, which controls and monitors one or more JMS listeners. Each listener monitors either a JMS queue destination (for point-to-point messaging) or a JMS topic destination (for publish/subscribe messaging). A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. When you deploy a message-driven bean, you associate the bean with a listener port. When a message arrives on the destination, the listener passes the message to a new instance of a message-driven bean for processing. For more information, see Message-driven beans - listener port components.

Procedure

1. Configure the message listener service.
2. Administer listener ports.

   You can complete any of the following administrative tasks:
   - Create or configure a listener port.
   - Start or stop a listener port.
   - Delete a listener port.
3. Configure security for message-driven beans that use listener ports.

Results

You have configured the resources needed by the message listener service to support message-driven beans.

Configuring the message listener service

To support message-driven beans deployed against listener ports, you configure the properties of the message listener service for an application server.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

The message listener service is an extension to the JMS functions of the JMS provider and provides a listener manager, which controls and monitors one or more JMS listeners. Each listener monitors either a JMS queue destination (for point-to-point messaging) or a JMS topic destination (for publish/subscribe messaging). For more information, see Message-driven beans - listener port components.

When you deploy an enterprise application to use message-driven beans with listener ports, you can browse or change the configuration of the message listener service for a given application server.

If your messaging system is running in non-ASF mode, to avoid unwanted transaction timeouts, you must make sure that the time that you specify for the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property is smaller than the sum of the maximum amount of time that the `onMessage()` method of the message-driven bean (MDB) takes to process the message, plus the time you specify for the **Total transaction lifetime timeout** transaction service property.

Procedure

1. Display the listener service settings page:
   a. In the navigation pane, select **Servers** > **Server Types** > **WebSphere application servers**.
   b. In the content pane, click the name of the application server.
   c. Under Communications, click **Messaging** > **Message Listener Service**.
2. Optional: Browse or change the value of properties for the message-driven bean thread pool.
   a. Click **Thread Pool**.
   b. Change the following properties, as required:

      Minimum size
      The minimum number of threads to allow in the pool.

      Maximum size
      The maximum number of threads to allow in the pool.

      Thread inactivity timeout
      The number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.

      Note: The administrative console does not allow you to set the inactivity timeout to a negative number. To do this you must modify the value directly in the config.xml file.

      Allow thread allocation beyond maximum thread size
      Select this check box to enable the number of threads to increase beyond the maximum size configured for the thread pool.

   c. Click **OK**.
3. Optional: Specify any message listener service custom properties that you need, as **Custom properties** of the message listener service.
   a. Click **Custom properties**
   b. For each custom property, specify the name and value you require.

      If you have not specified a property before:

      1) Click **New**.

2) Type the name of the property.
3) Type the value of the property.
4) Click **OK**.

For more information about these custom properties, see “Message listener service custom properties” on page 419.

4. Save your changes to the master configuration.
5. To activate the changed configuration, stop then restart the application server.

Results

You have configured the properties of the message listener service for a given application server.

Avoiding transaction timeouts in non-ASF mode

If your messaging system runs in non-Application Server Facilities (non-ASF) mode, you must configure the **Total transaction lifetime timeout** transaction service property and the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property correctly, to avoid unwanted transaction timeouts.

Before you begin

To carry out the steps in this task, your messaging system must be running in non-ASF mode. To change from ASF mode to non-ASF mode, add the **NON.ASF.RECEIVE.TIMEOUT** custom property to the message listener service as described in “Configuring the message listener service” on page 415.

About this task

For WebSphere Application Server Version 7 and later, listener ports are stabilized. For more information, read the article on stabilized features. You should plan to migrate your WebSphere MQ message-driven bean deployment configurations from using listener ports to using activation specifications. However, you should not begin this migration until you are sure the application does not have to work on application servers earlier than WebSphere Application Server Version 7. For example, if you have an application server cluster with some members at Version 6.1 and some at Version 7, you should not migrate applications on that cluster to use activation specifications until after you migrate all the application servers in the cluster to Version 7.

If your messaging system is running in non-ASF mode, to avoid unwanted transaction timeouts, you must make sure that the time that you specify for the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property is smaller than the sum of the maximum amount of time that the `onMessage()` method of the message-driven bean (MDB) takes to process the message, plus the time you specify for the **Total transaction lifetime timeout** transaction service property.

Procedure

1. To configure the **Total transaction lifetime timeout** transaction service property, complete step 8 in “Configuring transaction properties for an application server” on page 2491.
2. To configure the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service** > **Custom Properties**.
3. Click **NON.ASF.RECEIVE.TIMEOUT**. The General Properties page is displayed.
4. Modify the **Value** field. The value for **NON.ASF.RECEIVE.TIMEOUT** must be specified in milliseconds. Make sure that the value you specify, when converted into seconds (by dividing by 1000) , is less than the value you have specified for **Total transaction lifetime timeout** combined with the maximum number of seconds the onMessage() method of your MDB takes to process a message.
5. Click **OK**.

6. Stop and restart the application server.

Example

If **Total transaction lifetime timeout** and **NON.ASF.RECEIVE.TIMEOUT** are not correctly configured, transactions can time out before they are completed. This is because the thread begins calling the `receive()` method as soon as the transaction is created. In the following example, **NON.ASF.RECEIVE.TIMEOUT** is set to `110000` milliseconds (110 seconds), **Total transaction lifetime timeout** is set to `120` seconds and the onMessage () method of the MDB takes 15 seconds to process a message. The example supposes that a message does not appear at the destination until the `receive()` method has almost timed out:

1. The listener port starts and allocates a thread from the thread pool and creates a transaction on the thread.
2. The thread calls the `receive()` method to listen for messages.
3. After 110 seconds a message appears at the destination.
4. The thread removes the message from the destination and calls the `onMessage()` method of the MDB to begin processing the message.
5. Ten seconds later, the transaction timeout is reached. The application server marks the transaction for rollback.
6. Five seconds later, the `onMessage()` method finishes processing the message and tries to commit the transaction.
7. The total amount of time that has elapsed since the transaction was started is 125 seconds (110 seconds waiting for a message, plus 15 seconds to process the message). As this time is longer than the transaction timeout, the application server prevents the transaction from being committed, and it is rolled back.

Message listener service

The message listener service is an extension to the JMS functions of the JMS provider. It provides a listener manager that controls and monitors one or more JMS listeners, which each monitor a JMS destination on behalf of a deployed message-driven bean.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service**

Custom Properties

You can use the Custom properties page to define the following properties for use by the message listener service.

- "DYNAMIC.CONFIGURATION.ENABLED" on page 419
- "MAX.RECOVERY.RETRIES" on page 419
- "MQJMS.POOLING.THRESHOLD" on page 419
- "MQJMS.POOLING.TIMEOUT" on page 420
- "NON.ASF.RECEIVE.TIMEOUT" on page 420
- "NON.ASF.BMT.ROLLBACK.ENABLED" on page 421
- "RECOVERY.RETRY.INTERVAL" on page 421
- "SERVER.SESSION.POOL.REAP.TIME" on page 421
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT" on page 422
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*" on page 422

Listener Ports

You can use the Listener Ports page to create and modify listener ports by specifying the following properties:

- Name
- Initial State
- Description
- Connection factory JNDI name
- Destination JNDI name
- Maximum sessions
- Maximum retries
- Maximum messages

Thread pool

You can use the Thread pool page to change the following properties for the message-driven bean thread pool:

- Minimum size
- Maximum size
- Thread inactivity timeout

Message listener service custom properties:

Use this panel to view or change custom properties of the message listener service.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service** > **Custom Properties**.

You can use the Custom properties page to define the following properties for use by the message listener service.

- "DYNAMIC.CONFIGURATION.ENABLED" on page 419
- "MAX.RECOVERY.RETRIES" on page 419
- "MQJMS.POOLING.THRESHOLD" on page 419
- "MQJMS.POOLING.TIMEOUT" on page 420
- "NON.ASF.RECEIVE.TIMEOUT" on page 420
- "NON.ASF.BMT.ROLLBACK.ENABLED" on page 421
- "RECOVERY.RETRY.INTERVAL" on page 421
- "SERVER.SESSION.POOL.REAP.TIME" on page 421
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT" on page 422
- "SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*" on page 422

DYNAMIC.CONFIGURATION.ENABLED:

This property controls whether the application server on which a listener port is created requires to be restarted. Set this property to true to enable dynamic configuration.

Data type	Boolean
Default	False (not selected)

MAX.RECOVERY.RETRIES:

The maximum number of times that a listener port managed by this service tries to recover from a failure before giving up and stopping. When stopped the associated listener port is changed to the stop state. The interval between retry attempts is defined by the **RECOVERY.RETRY.INTERVAL** property.

A failure can be caused by either of the following conditions:

- An unexpected error has occurred when a listener port tries to get a message from the JMS provider.
- The connection between the application server and the JMS provider has been lost, usually due to a network error.

Data type	Integer
Units	Retry attempts
Default	5
Range	0 (no retries) through 2147483647

MQJMS.POOLING.THRESHOLD:

The maximum number of unused connections in the pool.

An MQSimpleConnectionManager allocates connections on a most-recently-used basis, and destroys connections on a least-recently-used basis. By default, a connection is destroyed if there are more than ten unused connections in the pool.

Data type	Integer
Units	Number of connections
Default	10

MQJMS.POOLING.TIMEOUT:

The number of milliseconds after which a connection in the pool is destroyed if it has not been used.

An MQSimpleConnectionManager allocates connections on a most-recently-used basis, and destroys connections on a least-recently-used basis. By default, a connection is destroyed if it has not been used for five minutes.

Data type	Integer
Units	Milliseconds
Default	5 minutes

NON.ASF.RECEIVE.TIMEOUT:

The timeout in milliseconds for synchronous message receives performed by message-driven bean listener sessions in the non-ASF mode of operation.

Note: The message listener service has two modes of operation, Application Server Facilities (ASF) and non-Application Server Facilities (non-ASF):

- ASF mode provides concurrency and transactional support for applications. For publish/subscribe message-drive beans, ASF mode provides better throughput and concurrency, because in the non-ASF mode the listener is single-threaded.

- Non-ASF mode is mainly for use with third-party messaging providers that do not support JMS ASF, which is an optional extension to the JMS specification. Non-ASF mode is also transactional but, because the path length is shorter than for ASF mode, usually provides improved performance.

To enable the non-ASF mode of operation for all message-driven bean listeners on the application server, set this property to a non-zero value.

If your messaging system is running in non-ASF mode, to avoid unwanted transaction timeouts, you must make sure that the time that you specify for the **NON.ASF.RECEIVE.TIMEOUT** message listener service custom property is smaller than the sum of the maximum amount of time that the `onMessage()` method of the message-driven bean (MDB) takes to process the message, plus the time you specify for the **Total transaction lifetime timeout** transaction service property.

Data type	Integer
Units	Milliseconds
Default	ASF mode (custom property not created)
Range	0 or greater milliseconds **0** Non-ASF mode is disabled **1 or more** The timeout in milliseconds for non-ASF message-driven bean listener synchronous session receives
Recommended	If a transaction timeout occurs, the message must recycle causing extra work. If you want to use the non-ASF mode, set this property to less than the transaction timeout, but greater than or equal to the maximum duration of your message-driven bean onMessage() method. For example, if your message-driven bean onMessage() method typically takes a maximum of 10 seconds, and the transaction timeout is set to 120 seconds, you might set the **NON.ASF.RECEIVE.TIMEOUT** property to no more than 110000 milliseconds (that is, 110 seconds).

NON.ASF.BMT.ROLLBACK.ENABLED:

When the non-Application Server Facilities (non-ASF) mode of operation is in use (because you have set the **NON.ASF.RECEIVE.TIMEOUT** property to a non-zero value), and a message-driven bean that uses bean-managed transactions generates a runtime exception, the **NON.ASF.BMT.ROLLBACK.ENABLED** property determines whether messages are returned to the destination.

Note: The message listener service has two modes of operation, Application Server Facilities (ASF) and non-Application Server Facilities (non-ASF):

- ASF mode provides concurrency and transactional support for applications. For publish/subscribe message-drive beans, ASF mode provides better throughput and concurrency, because in the non-ASF mode the listener is single-threaded.
- Non-ASF mode is mainly for use with third-party messaging providers that do not support JMS ASF, which is an optional extension to the JMS specification. Non-ASF mode is also transactional but, because the path length is shorter than for ASF mode, usually provides improved performance.

When this property is set to false (default), the message is automatically acknowledged before it is passed to the message-driven bean.

When this property is set to true, the message listener service sends a message acknowledgement to the client after the message is successfully processed by the message-driven bean, and the message listener service requests recovery of any message for which the bean generates an exception.

Data type	Boolean
Default	False

RECOVERY.RETRY.INTERVAL:

The time in seconds between retry attempts by a listener port to recover from a failure. The maximum number of retry attempts is defined by the **MAX.RECOVERY.RETRIES** property.

A failure can be caused by either of the following conditions:

- An unexpected error has occurred when a listener port tries to get a message from the JMS provider.
- The connection between the application server and the JMS provider has been lost, usually due to a network error.

Data type	Integer
Units	Seconds
Default	60
Range	1 through 2147483647

SERVER.SESSION.POOL.REAP.TIME:

The time in seconds between checks on server session pools. To enable server session pool monitoring, set this property to a non-negative value.

The **SERVER.SESSION.POOL.REAP.TIME** custom property is not applicable if your messaging system is running in non-ASF mode.

Data type	Integer
Units	Seconds
Default	-1 (disabled)
Range	-2147483648 through 2147483647

SERVER.SESSION.POOL.UNUSED.TIMEOUT:

The default server session pool timeout in seconds.

When this property is set to a non-negative value, it is compared to the time that has elapsed since a server session was used. If the timeout value is less than the elapsed time, the server session is removed from the server session pool and its JMS session is returned to the JMS session pool. For example, if the timeout value is one second and the time that has elapsed since a particular server session was used is two seconds, that server session is removed from the server session pool and its JMS session is returned to the JMS session pool.

The **SERVER.SESSION.POOL.UNUSED.TIMEOUT** custom property is not applicable if your messaging system is running in non-ASF mode.

Data type	Integer
Units	Seconds
Default	-1 (disabled)
Range	-2147483648 through 2147483647

SERVER.SESSION.POOL.UNUSED.TIMEOUT.lpname:

This property overrides the default **SERVER.SESSION.POOL.UNUSED.TIMEOUT** value for the listener port with the name defined for *lpname*. This value applies to all message-driven beans that use the specified listener port.

If this override is set to a non-negative value, it overrides the **SERVER.SESSION.POOL.UNUSED.TIMEOUT** property, even if the **SERVER.SESSION.POOL.UNUSED.TIMEOUT** property has a negative value.

If this override is set to a negative value, it disables server session pool monitoring for the specified listener port.

The **SERVER.SESSION.POOL.UNUSED.TIMEOUT.lpname** custom property is not applicable if your messaging system is running in non-ASF mode.

Data type	Integer
Units	Seconds
Default	Not set
Range	-2147483648 through 2147483647

Administering listener ports

You can use the WebSphere Application Server administrative console to administer listener ports, which each define the association between a connection factory, a destination, and a message-driven bean.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. If you set the initial state of a listener port to Started, the listener port is started automatically when a message-driven bean associated with that port is installed.

Listener ports can be manually started and stopped. If a message-driven bean fails to process a message several times, the listener port is automatically stopped by the application server. When a listener port is stopped, the listener manager stops the listeners for all message-driven beans associated with the port. Consequently, the associated message-driven beans can no longer process messages.

Note: You do not usually need to start or stop a listener port manually.

Procedure

- Create a new listener port.

Create a new listener port, to specify a new association between a connection factory, a destination, and a message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination.

- Configure a listener port.

  Browse or change the configuration properties of a listener port.
- Start a listener port.
- Stop a listener port.
- Delete a listener port.

Creating a new listener port

You create a new listener port for the message listener service to define the association between a connection factory, a destination, and a deployed message-driven bean. This association enables the message-driven bean to retrieve messages from the associated destination.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications.

If you have existing message-driven beans that use the WebSphere MQ messaging provider (or a compliant third-party JMS provider) with listener ports, and instead you want to use EJB 3 message-driven beans with listener ports, these new beans can continue to use the same messaging provider.

For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination. For more information, see Message-driven beans - listener port components.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**. The “Message listener port collection” on page 425 panel is displayed.
3. Click **New**.
4. Specify the following required properties:

   Name The name by which the listener port is known for administrative purposes.

   Connection factory JNDI name
   The JNDI name for the JMS connection factory to be used by the listener port; for example, `jms/connFactory1`.

   Destination JNDI name
   The JNDI name for the destination to be used by the listener port; for example, `jms/destn1`.
5. Optional: Change other properties for the listener port, as required.
6. Click **OK**.
7. Save your changes to the master configuration.

8. To have the changed configuration take effect, stop then restart the application server.

Results

If you set the initial state of a listener port to Started, the listener port is started automatically when a message-driven bean associated with that port is installed.

Configuring a listener port

Use this task to browse or change the properties of an existing listener port, which is used by message-driven beans associated with the port to retrieve messages.

Before you begin

If you want to use message-driven beans with a messaging provider that does not have a JCA 1.5 resource adapter, you cannot use activation specifications and therefore you must configure your beans against a listener port. There are also a few scenarios in which, although you could use activation specifications, you might still choose to use listener ports. For example, for compatability with existing message-driven bean applications. For more information about when to use listener ports rather than activation specifications, see Message-driven beans, activation specifications, and listener ports.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination. For more information, see Message-driven beans - listener port components.

When you deploy an enterprise application to use message-driven beans with listener ports, you can browse or change the configuration of a listener port.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.
3. Select the name of the listener port that you want to work with. This displays the properties of the listener port in the content pane.
4. Optional: Change properties for the listener port, according to your needs.
5. Click **OK**.
6. Save your changes to the master configuration.
7. To have a changed configuration take effect, stop then restart the application server.

Message listener port collection:

The message listener ports configured in the administrative domain

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination.

This panel displays a list of the message listener ports configured in the administrative domain. You can use this panel to add new listener ports or to change the properties of existing listener ports.

To view this administrative console panel, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Messaging] Message Listener Service** > **Listener Ports**

To manage a listener port, select the check box beside the listener port name in the list and click a button:

Button	Resulting action
Convert to activation specification	Opens a wizard that helps you convert the selected listener port to an activation specification.
New	Accesses the panel to configure a new listener port.
Delete	Deletes the selected listener port or ports.
Start	Starts the selected listener port or ports.
Stop	Stops the selected listener port or ports.

Listener port settings:

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. This association enables deployed message-driven beans associated with the port to retrieve messages from the destination.

Use this panel to view or change the configuration properties of the selected listener port.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Communications] Messaging** > **Message Listener Service** > **Listener Ports** > ***listener_port***.

Name:

The name by which the listener port is known for administrative purposes.

Data type String
Default Null

Initial state:

The state that you want the listener port to have when the application server is next restarted

Data type Enum
Units Not applicable
Default Started
Range

Started
When the application server is next started, the listener port is started automatically.

Stopped
When the application server is next started, the listener port is not started automatically. If message-driven beans are to use this listener port on the application server, the system administrator must start the port manually or select the Started value of this property then restart the application server.

Description:

A description of the listener port, for administrative purposes within IBM WebSphere Application Server.

Data type	String
Default	Null

Connection factory JNDI name:

The JNDI name for the JMS connection factory to be used by the listener port; for example, jms/connFactory1.

Data type	String
Default	Null

Destination JNDI name:

The JNDI name for the destination to be used by the listener port; for example, jms/destn1.

You cannot use a temporary destination for late responses.

Data type	String
Default	Null

Maximum sessions:

The maximum number of concurrent sessions that a listener can have with the JMS server to process messages.

Each session corresponds to a separate listener thread and therefore controls the number of concurrently processed messages. Adjust this parameter when the server does not fully use the available capacity of the machine.

Data type	Integer
Units	Sessions
Default	1
Range	1 through 2147483647
Recommended	• For message concurrency, that is to process multiple messages simultaneously, set this property to a value greater than 1. Keep this value as low as possible to prevent overloading client applications. A good starting point for a 100% JMS workload with short transaction times is 2 to 4 sessions per processor. If longer running transactions exist, you might need more sessions, which should be determined by experimentation. The total number of sessions specified in the Maximum Sessions property of all configured listener ports must be less than or equal to the number of threads specified for the Maximum Size property of the message listener service thread pool.

Maximum retries:

The maximum number of times that the listener tries to deliver a message to a message-driven bean instance before the listener is stopped, in the range 0 through 2147483647.

Note: A WebSphere MQ queue has a similar property called the **BackoutThreshold** property. If your listener port is reading from a WebSphere MQ queue, then the retry limit and the behavior when the limit is reached is determined by whichever of these two properties is set to the lower limit:

- If you exceed the WebSphere MQ queue **BackoutThreshold** limit, the message that cannot be delivered is moved to somewhere else by WebSphere MQ (for example, to the WebSphere MQ backout requeue queue or the WebSphere MQ dead letter queue) and the listener port services the next message on the queue. In this case, WebSphere Application Server might not know that the message has not been delivered successfully.
- If you exceed the listener port **maximum retries** limit, the listener port stops. You then manually intervene to investigate the problem, possibly to remove the message from the WebSphere MQ queue then restart the listener port.

Data type	Integer
Units	Retry attempts
Default	0 (no retries)
Range	0 (no retries) through 2147483647

Maximum messages:

The maximum number of messages that the listener can process in one transaction.

If the queue is empty, the listener processes each message when it arrives. Each message is processed within a separate transaction.

For the WebSphere V5 default messaging provider or WebSphere MQ as the JMS provider, if messages start accumulating on the queue then the listener can start processing messages in batches. For third-party messaging providers, this property value is passed to the JMS provider but the effect depends on the JMS provider.

Data type	Integer
Units	Number of messages
Default	1
Range	1 through 2147483647
Recommended	For the WebSphere default messaging providers or WebSphere MQ as the JMS provider, to process multiple messages in a single transaction, set this value to more than 1. If messages start accumulating on the queue, a value greater than 1 enables multiple messages to be batch-processed into a single transaction, and eliminates much of the transaction processing costs for JMS messages. **CAUTION:** • If one message in the batch fails processing with an exception, the entire batch of messages is put back on the queue for processing. • Any resource lock held by any of the interactions for the individual messages are held for the duration of the entire batch. • Depending on the amount of processing that messages need, and if XA transactions are being used, setting a value greater than 1 can cause the transaction to time out. If an XA transaction does time out routinely because processing multiple messages exceeds the transaction timeout, reduce this property to 1 (to limit processing to one message per transaction) or increase your transaction timeout.

Starting a listener port

Use this task to start a listener port on an application server, to enable the listeners for message-driven beans associated with the port to retrieve messages.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. For more information, see Message-driven beans - listener port components.

A listener is active, that is able to receive messages from a destination, if the deployed message-driven bean, listener port, and message listener service are all started. Although you can start these components in any order, they must all be in a started state before the listener can retrieve messages.

If you set the initial state of a listener port to Started, the listener port is started automatically when a message-driven bean associated with that port is installed. You can also start a listener port manually.

When a listener port is started, the listener manager tries to start the listeners for each message-driven bean associated with the port. If a message-driven bean is stopped, the port is started but the listener is not started, and remains stopped. If you start a message-driven bean, the related listener is started.

Procedure

1. Start the administrative console.
2. If you want the listener for a deployed message-driven bean to be able to receive messages at the port, check that the message-driven bean has been started.
3. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.
4. Select the check box for the listener port that you want to start.
5. Click **Start**.
6. Save your changes to the master configuration.

Stopping a listener port

Use this task to stop a listener port on an application server, to prevent the listeners for message-driven beans associated with the port from retrieving messages.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. For more information, see Message-driven beans - listener port components.

If a message-driven bean fails to process a message several times, the listener port is automatically stopped by the application server. You can also stop a listener port manually. When a listener port is stopped, the listener manager stops the listeners for all message-driven beans associated with the port. Consequently, the associated message-driven beans can no longer process messages.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.

3. Select the check box for the listener port that you want to stop.
4. Click **Stop**.
5. Save your changes to the master configuration.
6. To have the changed configuration take effect, stop then restart the application server.

Deleting a listener port

Use this task to delete a listener port from the message listener service, to prevent message-driven beans associated with the port from retrieving messages.

About this task

A listener port defines the association between a connection factory, a destination, and a deployed message-driven bean. For more information, see Message-driven beans - listener port components.

To delete a listener port, use the administrative console to complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers->*server_name*** > **[Communications] Messaging** > **Message listener service** > **[Additional Properties] Listener Ports**

   The “Message listener port collection” on page 425 panel is displayed.
3. Select the check box for the listener port that you want to delete.
4. Click **Delete**.

   This action stops the port (needed to allow the port to be deleted) then deletes the port.
5. Save your changes to the master configuration.
6. To have the changed configuration take effect, stop then restart the application server.

Monitoring server session pools for listener ports

You can minimize the number of resources that server sessions use by enabling server session pool monitoring and defining the timeout value to be applied to a server session.

About this task

Each listener port uses one or more server sessions, which are held in a server session pool. Each server session is associated with a JMS session, which is taken from the JMS session pool that is associated with the JMS connection factory that the listener port is configured to use.

By default, server session pool monitoring is disabled. When a listener port uses a server session the listener port does not release the server session from the server session pool until the listener port is shut down. This means that the associated JMS session is not released into the JMS session pool until the listener port is shut down, even if the listener port is not processing any messages. Consequently the resources that the JMS session uses, for example TCP/IP connections, can be held for a long time, and this can cause problems for resource-constrained systems.

To minimize the number of resources that server sessions use, you must monitor the server session pools. When you enable server session pool monitoring each server session in each server session pool that a listener port uses is monitored to determine how much time has elapsed since the server session was last used. If the elapsed time is greater than the timeout value that you have configured, the server session is removed from the server session pool and its associated JMS session is returned to the JMS session pool. The returned JMS session can be either reused by another application or closed, depending on your JMS session pool settings. You can also configure additional pooling mechanisms, depending on your JMS provider.

Note: Server session pool monitoring cannot be used if the message listener service is operating in non-Application Server Facilities (non-ASF) mode, that is if the NON.ASF.RECEIVE.TIMEOUT message listener service custom property is set to a non-zero value.

Procedure

To enable server session pool monitoring, configure the following message listener service custom properties on each application server as required.

SERVER.SESSION.POOL.REAP.TIME
To enable server session pool monitoring, set this property to the time in seconds between checks on server session pools (this must be a non-negative value).

SERVER.SESSION.POOL.UNUSED.TIMEOUT
To specify the default server session pool timeout, set this property to the required number of seconds for the timeout. When this property is set to a non-negative value, it is compared with the time that has elapsed since a server session was used. If the timeout value is less than the elapsed time, the server session is removed from the server session pool and its JMS session is returned to the JMS session pool. For example, if the timeout value is one second and the time that has elapsed since a particular server session was used is two seconds, that server session is removed from the server session pool and its JMS session is returned to the JMS session pool.

SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*
To override the default SERVER.SESSION.POOL.UNUSED.TIMEOUT value for the listener port with the name *lpname*, set this property to the appropriate value:

- To override the SERVER.SESSION.POOL.UNUSED.TIMEOUT for the specified listener port, set this property to a non-negative value defining the required number of seconds for the server session timeout for this listener port.
- To disable server session pool monitoring for the specified listener port, set this property to a negative value.

The value that you set for this property applies to all message-driven beans that are using the specified listener port.

Example

For example, consider an application server that is configured with listener ports lp1, and lp2.

The following rules apply:

No properties set
If none of the properties are set, server session pool monitoring is disabled and JMS sessions used by server sessions are not returned to the JMS session pool until the listener port (lp1 or lp2), or its associated message-driven bean, is shut down.

SERVER.SESSION.POOL.REAP.TIME and SERVER.SESSION.POOL.UNUSED.TIMEOUT set
Consider, for example, the following settings:

SERVER.SESSION.POOL.REAP.TIME=60

SERVER.SESSION.POOL.UNUSED.TIMEOUT=120

The server session pool of both listener ports (lp1 and lp2) is checked for inactive server sessions every 60 seconds. If a server session is detected as being inactive for more than 120 seconds, it is removed from the server session pool and its JMS session is returned to the JMS session pool. Taking into account the SERVER.SESSION.POOL.REAP.TIME value, the server session pool could be removed from the session pool between two and three minutes after the server session was last used.

SERVER.SESSION.POOL.REAP.TIME and SERVER.SESSION.POOL.UNUSED.TIMEOUT set, and overrides set for SERVER.SESSION.POOL.UNUSED.TIMEOUT.*lpname*

Consider, for example, the following settings:

```
SERVER.SESSION.POOL.REAP.TIME=60
SERVER.SESSION.POOL.UNUSED.TIMEOUT=120
SERVER.SESSION.POOL.UNUSED.TIMEOUT.lp2=-1
SERVER.SESSION.POOL.UNUSED.TIMEOUT.lp1=60
```

The server session pool for listener port lp2 is not checked because it has a negative timeout value. In the server session pool for listener port lp1, any server sessions that are inactive for more than 60 seconds are removed from the server session pool.

Chapter 16. Administering Naming and directory

This page provides a starting point for finding information about naming support. Naming includes both server-side and client-side components. The server-side component is a Common Object Request Broker Architecture (CORBA) naming service (CosNaming). The client-side component is a Java™ Naming and Directory Interface (JNDI) service provider. JNDI is a core component in the Java Platform, Enterprise Edition (Java EE) programming model.

The WebSphere® JNDI service provider can be used to interoperate with any CosNaming name server implementation. Yet WebSphere name servers implement an extension to CosNaming, and the JNDI service provider uses those WebSphere extensions to provide greater capability than CosNaming alone. Some added capabilities are binding and looking up of non-CORBA objects.

Java EE applications use the JNDI service provider supported by WebSphere Application Server to obtain references to objects related to server applications, such as enterprise bean (EJB) homes, which have been bound into a CosNaming name space.

Configuring namespace bindings

Instead of creating namespace bindings from a program, you can configure namespace bindings using the administrative console. Name servers add these configured bindings to the namespace view by reading the configuration data for the bindings. Configured bindings are created each time a server starts, even when the binding is created in a transient partition of the namespace. One major use of configured bindings is to provide fixed qualified names for server application objects.

Before you begin

Assemble and deploy your application onto an application server. If the application is a client to an application running in another server process, specify qualified jndiName values for the server objects of the other application during assembly or deployment. For more information on qualified names, refer to the topic on lookup names support in deployment descriptors and thin clients.

About this task

A deployed application requires qualified fixed names if the application is accessed by thin client applications or by Java Platform, Enterprise Edition (Java EE) client applications or server applications running in another server process.

When you configure a namespace binding, you create a qualified fixed name for a server object. A fixed name does not change if the object is moved to another server. A qualified fixed name with a cell scope has the following form:

`cell/persistent/`*`fixedName`*

The *fixedName* is an arbitrary fixed name.

You can configure namespace bindings, and thus qualified fixed names, for the following objects:
- A string constant value
- An enterprise bean (EJB) home installed on a server in the cell
- A CORBA object available from a CosNaming name server
- An object bound in a WebSphere Application Server namespace that is accessible using a Java Naming and Directory Interface (JNDI) indirect lookup

To view or configure a namespace binding for an object of a deployed application, complete the following:

Procedure

1. Go to the Name space bindings page.

   In the administrative console, click **Environment** > **Naming** > **Name space bindings**.

2. Select the desired scope.

   The scope determines where in the namespace binding is created. It also affects which name servers contain the binding in the namespace that they manage. Regardless of the scope, a namespace binding is accessible from all name servers in the cell. However, the scope can affect whether the lookup can be resolved locally by a name server or whether the name server must make a remote call to another name server to resolve the binding.

   Only namespace bindings created with the selected scope are visible in the collection table on the page. By changing the scope, you can see and create bindings in other scopes.

   a. Select a scope.

      If you are creating a new namespace binding, refer to the table below as a guide in selecting a scope:

Table 66. Namespace binding scope descriptions. The scope can be a cell, node, server, or cluster.

Scope	Description
Cell	Cell-scoped bindings are created under the cell persistent root context. Select **Cell** if the namespace binding is not specific to any particular node or server, or if you do not want the binding to be associated with any specific node or server. For example, you can use cell-scoped bindings to create fixed qualified names for enterprise beans. Fixed qualified names do not have any node or server names embedded within them.
Node	Node-scoped bindings are created under the node persistent root context for the selected node. Select **Node** if the namespace binding is specific to a particular node, or if you want the binding to be associated with a specific node.
Server	Server-scoped bindings are created under the server root context for the selected server. Select **Server** if a binding is to be used only by clients of an application running on a particular server, or if you want to configure a binding with the same name on different servers which resolve to different objects. Note that two servers can have configured bindings with the same name but resolve to different objects. Server-scoped bindings are created in the process of the selected application server. Therefore, the name server running in the selected application server can resolve those bindings locally. No remote invocations to other name servers are necessary to resolve the bindings. However, all other name servers in the cell must make remote calls to the selected server in order to resolve the bindings. For example, in order for the name server running in *server1* in node *node1* to resolve the name `cell/nodes/node1/servers/server2/serverScopedConfiguredBinding`, it must make a remote call to *server2* in *node1*. Only the name server in *server2* in *node1* can resolve that name without invoking any other name servers.

   b. Click **Apply**.

3. Create a new namespace binding.

   a. Open the New Name Space Binding wizard.

      On the Name space bindings page, click **New**.

   b. On the **Specify binding type** page, select the binding type.

      The namespace binding can be for a constant string value, an EJB home, a CORBA CosNaming NamingContext or CORBA leaf node object, or an object that you can look up indirectly using JNDI.

   c. On the **Specify basic properties** page, specify the binding identifier and other properties for the binding.

      For property descriptions, refer to the following:
      - "String binding settings" on page 1019
      - "EJB binding settings" on page 1020
      - "CORBA object binding settings" on page 1021
      - "Indirect lookup binding settings" on page 1022

d. On the **Summary** page, verify the settings and click **Finish**.

The name of the new binding is displayed in the collection table on the Name space bindings page.

4. Optional: Edit a previously created binding.
   a. From the collection table on the Name space bindings page, click the name of the binding that you want to edit.
   b. Edit the binding properties as desired. Step 3(c) provides links to property descriptions.
   c. Click **OK**.

Results

Cell-scoped bindings are created under the cell persistent root context. Node-scoped bindings are created under the node persistent root context for the specified node. Server-scoped bindings are created under the server root context for the selected server.

Name space binding collection

Use this page to configure a name binding of an EJB, a CORBA CosNaming NamingContext, a CORBA leaf node object, an object that you can look up using Java Naming and Directory Interface (JNDI), or a constant string value.

Binding information for configured bindings is stored in the configuration and applied upon startup of the name server for each server within the scope of the binding.

To view the Name space bindings page, click **Environment** > **Naming** > **Name space bindings**.

Click the check boxes to select one or more of the bindings in your collection. Use the buttons to control the selected bindings.

When creating a new binding, select a scope before clicking **New**. The **Scope** setting filters what bindings are listed in the collection as well as sets the scope of the new binding.

Name

Shows the names given to uniquely identify these configured bindings.

Scope

Shows the scope of the configured binding. This value indicates the configuration location for the `namebindings.xml` file. This field is for information purposes only and cannot be updated.

If the configured binding is cell-scoped, the starting context is the cell persistent root context. If the configured binding is node-scoped, the starting context is the node persistent root context. If the configured binding is server-scoped, the starting context is the server's server root context.

Binding type

Shows the type of binding configured. Valid values are String, EJB, CORBA, and Indirect. This field is for information purposes only and cannot be updated.

Specify binding type settings

Use this page to select the type of namespace binding that you want.

To view this administrative console page, click **Environment** > **Naming** > **Name space bindings** > **New**.

You can configure a namespace binding for any of the following objects:

- A string constant value
- An enterprise bean (EJB) home installed on a server in the cell

- A CORBA object available from a CosNaming name server
- An object bound in a WebSphere Application Server namespace that is accessible using a Java Naming and Directory Interface (JNDI) indirect lookup

On this page, select a binding type and then click **Next**.

Binding type

Specifies the type of binding configured.

Table 67. Namespace binding types. Available types include String, EJB, CORBA, or Indirect.

String	Select **String** to configure a namespace binding for a string constant value. To configure a String binding, you need the following information: • The string constant value • The target root context for the configured binding (scope) • The name of the configured binding, relative to the target root context You can create a file that maps multiple variable names to values and specify the file name for the **String** value. By default, a name server performs variable substitution on the string value of a String namespace binding. Thus, by default, the com.ibm.websphere.naming.expandStringBindings property is set to `true` and a name server expands the value of String bindings. **Tip:** Variable substitution can result in errors or unexpected changes to a string. For example, with variable substitution, a `$$` string is expanded as `$`. You can disable variable substitution and cause the name server to treat the **String** value as a literal or constant. Create a custom property with **Name** set to `com.ibm.websphere.naming.expandStringBindings` and **Value** set to `false`. You can define a custom property at the cell, node, server, or name server scope. Create the custom property on a console page for the appropriate scope: **Cell scope** Click **System administration** > **Cell** > **Custom properties** > **New**. **Node scope** Click **System administration** > **Nodes** > ***node_name*** > **Custom properties** > **New**. **Server scope** Click **Application servers** > ***server_name*** > **Administration** > **Custom properties** > **New**. **Name server scope** Click **Application servers** > ***server_name*** > **Administration** > **Server components** > **Name server** > **Custom properties** > **New**. All name servers within the specified custom property scope apply the setting. Settings at a narrower scope override settings at a wider scope. For example, on multiple-server products, settings at the node scope override settings at the cell scope. Select a custom property scope that is at least as wide as the namespace binding scope. Thus, to prevent variable expansion in a cell-scoped String namespace binding, define the custom property at the cell scope. If the custom property has a scope narrower than the namespace binding, only name servers within the scope prevent variable expansion in the String namespace binding. Name servers outside of the scope expand the variable reference and handle the reference differently.
EJB	Select **EJB** to configure a namespace binding for an EJB home installed on a server in the cell. Use a cell-scoped EJB binding to create a fixed qualified lookup name for an enterprise bean. A fixed qualified lookup name is not dependent on the cell topology. To configure an EJB home binding, you need the following information: • The JNDI name of the EJB server or server cluster where the enterprise bean is deployed • The target root context for the configured binding (scope) • The name of the configured binding, relative to the target root context On stand-alone servers, do not configure an EJB binding that resolves to another server. The name server cannot read configuration data for other servers. That data is required to construct the binding.

Table 67. Namespace binding types (continued). Available types include String, EJB, CORBA, or Indirect.

CORBA	Select **CORBA** to configure a namespace binding for a Common Object Request Broker: Architecture and Specification (CORBA) object available from a Object Management Group (OMG) Interoperable Naming (CosNaming) name server. Identify a CORBA object bound into an INS compliant CosNaming server with a corbaname URL. The referenced object does not have to be available until the binding is actually referenced by an application. To configure a CORBA binding, you need the following information: • The corbaname URL of the CORBA object • An indicator if the bound object is a context or leaf node object (to set the correct CORBA binding type of context or object) • The target root context for the configured binding • The name of the configured binding, relative to the target root context
Indirect	Select **Indirect** to configure a namespace binding for an object bound in a WebSphere Application Server namespace that is accessible using a JNDI indirect lookup. You can select **Indirect** for CORBA objects as well as for javax.naming.Referenceable, javax.naming.Reference, and java.io.Serializable objects. The target object itself is not bound to the namespace. Only the information required to look up the object is bound. Therefore, the referenced name server does not have to be running until the binding is actually referenced by some application. To configure an indirect JNDI lookup binding, you need the following information: • The JNDI provider URL of the name server where the object resides • The JNDI lookup name of the object • The target root context for the configured binding (scope) • The name of the configured binding, relative to the target root context The following information is optional: • The JNDI initial context factory class name. The default is the WebSphere Application Server initial context factory, com.ibm.websphere.naming.WsnInitialContextFactory. • Additional properties to pass to the javax.naming.InitialContext constructor. A cell-scoped indirect binding is useful when creating a fixed qualified lookup name for a bound object so that the qualified lookup name is not dependent on the cell topology.

String binding settings

Use this page to view or configure a string binding.

To view this administrative console page, click **Environment** > **Naming** > **Name space bindings** > ***string_namespace_binding***. The settings on this page are similar to those on the **Specify basic properties** panel of the New name space binding wizard.

Scope

Shows the scope of the configured binding. This value indicates the configuration location for the `namebindings.xml` file.

The **Scope** setting is shown only when you edit an existing binding on the console page, and is not shown when you create a new binding on the wizard panel. This setting is for information purposes and cannot be updated.

If the configured binding is cell-scoped, the starting context is the cell persistent root context. If the configured binding is node-scoped, the starting context is the node persistent root context. If the configured binding is server-scoped, the starting context is the server's server root context.

Binding type
Shows the type of binding configured. Possible choices are String, EJB, CORBA, and Indirect. This setting is for information purposes only and cannot be updated.

Binding identifier
Specifies the name that uniquely identifies this configured binding.

Name in name space
Specifies the name used for this binding in the namespace. This name can be a simple or compound name depending on the portion of the namespace where this binding is configured.

String value
Specifies the string to be bound into the namespace.

You can create a file that maps multiple variable names to values and specify the file name for the **String** value. By default, a name server performs variable substitution on the string value of a String namespace binding. Thus, by default, the com.ibm.websphere.naming.expandStringBindings property is set to `true` and a name server expands the value of String bindings.

Tip: Variable substitution can result in errors or unexpected changes to a string. For example, with variable substitution, a `$$` string is expanded as `$`. You can disable variable substitution and cause the name server to treat the **String** value as a literal or constant. Create a custom property with **Name** set to `com.ibm.websphere.naming.expandStringBindings` and **Value** set to `false`. You can define a custom property at the cell, node, server, or name server scope. Create the custom property on a console page for the appropriate scope:

Cell scope
Click **System administration** > **Cell** > **Custom properties** > **New**.

Node scope
Click **System administration** > **Nodes** > ***node_name*** > **Custom properties** > **New**.

Server scope
Click **Application servers** > ***server_name*** > **Administration** > **Custom properties** > **New**.

Name server scope
Click **Application servers** > ***server_name*** > **Administration** > **Server components** > **Name server** > **Custom properties** > **New**.

All name servers within the specified custom property scope apply the setting. Settings at a narrower scope override settings at a wider scope. For example, on multiple-server products, settings at the node scope override settings at the cell scope. Select a custom property scope that is at least as wide as the namespace binding scope. Thus, to prevent variable expansion in a cell-scoped String namespace binding, define the custom property at the cell scope. If the custom property has a scope narrower than the namespace binding, only name servers within the scope prevent variable expansion in the String namespace binding. Name servers outside of the scope expand the variable reference and handle the reference differently.

EJB binding settings
Use this page to configure a new enterprise bean (EJB) binding, or to view or edit an existing EJB binding.

To view this console page, click **Environment** > **Naming** > **Name space bindings** > ***EJB_namespace_binding***. The settings on this page are similar to those on the **Specify basic properties** panel of the New name space binding wizard.

Scope
Shows the scope of the configured binding. This value indicates the configuration location for the `namebindings.xml` file.

The **Scope** setting is shown only when you edit an existing binding on the console page, and is not shown when you create a new binding on the wizard panel. This setting is for information purposes and cannot be updated.

If the configured binding is cell-scoped, the starting context is the cell persistent root context. If the configured binding is node-scoped, the starting context is the node persistent root context. If the configured binding is server-scoped, the starting context is the server's server root context.

Binding type
Shows the type of binding configured. Possible choices are String, EJB, CORBA, and Indirect. This setting is for information purposes only and cannot be updated.

Binding identifier
Specifies the name that uniquely identifies this configured binding.

Name in name space
Specifies the name used for this binding in the namespace. This name can be a simple or compound name depending on the portion of the namespace where this binding is configured.

Enterprise bean location
Specifies whether the enterprise bean is running in a single server. If `Single server` is specified, type the node name.

Server
Specifies the name of the server in which the enterprise bean is configured.

JNDI name
Specifies the Java Naming and Directory Interface (JNDI) name of the deployed enterprise bean. It is the bean's JNDI name that is in the enterprise bean bindings, not the java:comp name.

CORBA object binding settings
Use this page to configure a new name binding of a CORBA object binding, or to view or edit an existing CORBA object binding.

To view this console page, click **Environment** > **Naming** > **Name space bindings** > ***CORBA_namespace_binding***. The settings on this page are similar to those on the **Specify basic properties** panel of the New name space binding wizard.

Scope
Shows the scope of the configured binding. This value indicates the configuration location for the `namebindings.xml` file.

The **Scope** setting is for information purposes and cannot be updated.

If the configured binding is cell-scoped, the starting context is the cell persistent root context. If the configured binding is node-scoped, the starting context is the node persistent root context. If the configured binding is server-scoped, the starting context is the server's server root context.

Binding type
Shows the type of binding configured. Possible choices are String, EJB, CORBA, and Indirect. This setting is for information purposes and cannot be updated.

Binding identifier
Specifies the name that uniquely identifies this configured binding.

Name in name space
Specifies the name used for this binding in the namespace. This name can be a simple or compound name depending on the portion of the namespace where this binding is configured.

Corbaname URL
Specifies the CORBA name URL string identifying where the object is bound in a CosNaming server.

Federated context
Specifies whether the target is a CosNaming context (true) or a leaf node object (false).

true	The target object is bound with a context CORBA binding type. If the corbaname URL does not resolve to a NamingContext, an error occurs when the binding is first used (which is when the URL is first resolved).
false	The target object is bound with an object CORBA binding type.

Indirect lookup binding settings
Use this page to configure a new indirect lookup name binding, or to view or edit an existing indirect lookup binding.

To view this console page, click **Environment** > **Naming** > **Name space bindings** > ***indirect_lookup_namespace_binding***. The settings on this page are similar to those on the **Specify basic properties** panel of the New name space binding wizard.

Scope
Shows the scope of the configured binding. This value indicates the configuration location for the `namebindings.xml` file.

The **Scope** setting is shown only when you edit an existing binding on the console page, and is not shown when you create a new binding on the wizard panel. This setting is for information purposes and cannot be updated.

If the configured binding is cell-scoped, the starting context is the cell persistent root context. If the configured binding is node-scoped, the starting context is the node persistent root context. If the configured binding is server-scoped, the starting context is the server's server root context.

Binding type
Shows the type of binding configured. Possible choices are String, EJB, CORBA, and Indirect. This setting is for information purposes only and cannot be updated.

Binding identifier
Specifies the name that uniquely identifies this configured binding.

Name in name space
Specifies the name used for this binding in the namespace. This name can be a simple or compound name depending on the portion of the namespace where this binding is configured.

Provider URL
Specifies the provider URL string needed to obtain a Java Naming and Directory Interface (JNDI) initial context.

JNDI name
Specifies the name used to look up the target object from the initial context.

Initial context factory name
Specifies the class name of the initial context factory used to obtain a JNDI initial context.

This field is optional. If no factory is specified, the WebSphere Application Server initial context factory is used.

If the scope of the indirect lookup binding includes servers or nodes previous to Version 6.1, the initial context factory name and other context factory properties are not shown on the console page.

Configuring name servers

Name servers add configured name space bindings to the name space view by reading the configuration data for the bindings. If you use configured bindings, you do not need to create name space bindings from a program.

About this task

You can configure a name server to provide a display name and initial state for an application server, as well as specify custom properties for the name server. Configure a name server using the administrative console.

Procedure
1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Administration** > **Server components** > **Name server**.
2. Edit the fields as needed.
   All of the fields are mandatory.
3. To make other changes, click **Custom properties** and configure a custom property.
4. Click **OK** to register your changes.

What to do next

Examine and start the name server.

Name server settings
Use this page to configure Naming Service Provider settings for the application server.

To view this administrative console page, click one of the following paths:
- **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Administration** > **Server components** > **Name server**
- **Servers** > **Server Types** > **Version 5 JMS servers** > ***server_name*** > **Administration** > **Server components** > **Name server**

Name
Specifies the display name for the server component.

Data type String

Initial state
Specifies the execution state. The options are: *Started* and *Stopped*.

Data type String
Default Started

Chapter 17. Administering Object pools

This page provides a starting point for finding information about object pools.

Object pools provide an effective means of improving application performance at run time, by supporting the reuse of multiple instances of objects.

Using object pools

An object pool helps an application avoid creating new Java objects repeatedly. Most objects can be created once, used and then reused. An object pool supports the pooling of objects waiting to be reused.

About this task

Object pools are not meant to be used for pooling JDBC connections or Java Message Service (JMS) connections and sessions. WebSphere Application Server provides specialized mechanisms for dealing with those types of objects. These object pools are intended for pooling application-defined objects or basic Developer Kit types.

To use an object pool, the product administrator must define an *object pool manager* using the administrative console. Multiple object pool managers can be created in an Application Server cell.

Note: The Object pool manager service is only supported from within the EJB container or Web container. Looking up and using a configured object pool manager from a Java 2 Platform Enterprise Edition (J2EE) application client container is not supported.

Procedure

1. Start the administrative console.
2. Click **Resources > Object pool managers**.
3. Specify a **Scope** value and click **New**.
4. Specify the required properties for work manager settings.
   Scope The scope of the configured resource. This value indicates the location for the configuration file.
   Name The name of the object pool manager. This name can be up to 30 ASCII characters long.
   JNDI Name
   The Java Naming and Directory Interface (JNDI) name for the pool manager.
5. [Optional] Specify a **Description** and a **Category** for the object pool manager.

Results

After you have completed these steps, applications can find the object pool manager by doing a JNDI lookup using the specified JNDI name.

Example

The following code illustrates how an application can find an object pool manager object:

```
InitialContext ic = new InitialContext();
ObjectPoolManager opm = (ObjectPoolManager)ic.lookup("java:comp/env/pool");
```

When the application has an ObjectPoolManager, it can cache an object pool for classes of the types it wants to use. The following is an example:

```
ObjectPool arrayListPool = null;
ObjectPool vectorPool = null;
try
```

```
{
 arrayListPool = opm.getPool(ArrayList.class);
 vectorPool = opm.getPool(Vector.class);
}
catch(InstantiationException e)
{
 // problem creating pool
}
catch(IllegalAccessException e)
{
 // problem creating pool
}
```

When the application has the pools, the application can use them as in the following example:

```
ArrayList list = null;
try
{
 list = (ArrayList)arrayListPool.getObject();
 list.clear(); // just in case
 for(int i = 0; i < 10; ++i)
 {
  list.add("" + i);
 }
 // do what ever we need with the ArrayList
}
finally
{
 if(list != null) arrayListPool.returnObject(list);
}
```

This example presents the basic pattern for using object pooling. If the application does not return the object, then the only adverse effect is that the object cannot be reused.

Object pool managers

Object pool managers control the reuse of application objects and Developer Kit objects, such as Vectors and HashMaps.

Multiple object pool managers can be created in an Application Server cell. Each object pool manager has a unique cell-wide Java Naming and Directory Interface (JNDI) name. Applications can find a specific object pool manager by doing a JNDI lookup using the specific JNDI name.

The object pool manager and its associated objects implement the following interfaces:

```
public interface ObjectPoolManager
{
 ObjectPool getPool(Class aClass)
  throws InstantiationException, IllegalAccessException;
 ObjectPool createFastPool(Class aClass)
  throws InstantiationException, IllegalAccessException;

}

public interface ObjectPool
{
 Object getObject();
 void returnObject(Object o);

}
```

The getObject() method removes the object from the object pool. If a getObject() call is made and the pool is empty, then an object of the same type is created. A returnObject( ) call puts the object back into the object pool. If returnObject() is not called, then the object is no longer allocatable from the object pool. If the object is not returned to the object pool, then it can be garbage collected.

Each object pool manager can be used to pool any Java object with the following characteristics:
- The object must be a public class with a public default constructor.
- If the object implements the java.util.Collection interface, it must support the optional clear() method.

Each pooled object class must have its own object pool. In addition, an application gets an object pool for a specific object using either the ObjectPoolManager.getPool() method or the ObjectPoolManager.createFastPool() method. The difference between these methods is that the getPool() method returns a pool that can be shared across multiple threads. The createFastPool() method returns a pool that can only be used by a single thread.

If in a Java virtual machine (JVM), the getPool() method is called multiple times for a single class, the same pool is returned. A new pool is returned for each call when the createFastPool() method is called. Basically, the getPool() method returns a pool that is thread-synchronized.

The pool for use by multiple threads is slightly slower than a fast pool because of the need to handle thread synchronization. However, extreme care must be taken when using a fast pool.

Consider the following interface:

```
public interface PoolableObject
{
 void init();
 void returned();
}
```

If the objects placed in the pool implement this interface and the ObjectPool.getObject() method is called, the object that the pool distributes has the init() method called on it. When the ObjectPool.returnObject() method is called, the PoolableObject.returned() method is called on the object before it is returned to the object pool. Using this method objects can be pre-initialized or cleaned up.

It is not always possible for an object to implement PoolableObject. For example, an application might want to pool ArrayList objects. The ArrayList object needs clearing each time the application reuses it. The application might extend the ArrayList object and have the ArrayList object implement a poolable object. For example, consider the following:

```
public class PooledArrayList extends ArrayList implements PoolableObject
{
 public PooledArrayList()
 {
 }

 public void init() {
 }

 public void returned()
 {
  clear();
 }
}
```

If the application uses this object, in place of a true ArrayList object, the ArrayList object is cleared automatically when it is returned to the pool.

Clearing an ArrayList object simply marks it as empty and the array backing the ArrayList object is not freed. Therefore, as the application reuses the ArrayList, the backing array expands until it is big enough for all of the application requirements. When this point is reached, the application stops allocating and copying new backing arrays and achieves the best performance.

It might not be possible or desirable to use the previous procedure. An alternative is to implement a custom object pool and register this pool with the object pool manager as the pool to use for classes of

that type. The class is registered by the WebSphere administrator when the object pool manager is defined in the cell. Take care that these classes are packaged in Java Archive (JAR) files available on all of the nodes in the cell where they might be used.

Object pool managers collection

An object pool manages a pool of arbitrary objects and helps applications avoid creating new Java objects repeatedly. Most objects can be created once, used and then reused. An object pool supports the pooling of objects waiting to be reused. These object pools are not meant to be used for pooling Java Database Connectivity connections or Java Message Service (JMS) connections and sessions. WebSphere Application Server provides specialized mechanisms for dealing with those types of objects. These object pools are intended for pooling application-defined objects or basic Developer Kit types.

To view this administrative console page, click **Resources > Object pool managers**.

To use an object pool, the product administrator must define an object pool manager using the administrative console. Multiple object pool managers can be created in an Application Server cell.

Name

Specifies the name by which the object pool manager is known for administrative purposes.

Data type String
Range 1 through 30 ASCII characters

JNDI name

Specifies the Java Naming and Directory Interface (JNDI) name for the object pool manager.

Data type String

Scope

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Description

Specifies the description of the object pool manager.

Data type String

Category

Specifies the category name used to classify or group this object pool manager.

Data type String

Object pool managers settings

An object pool manages a pool of arbitrary objects and helps applications avoid creating new Java objects repeatedly. Most objects can be created once, used and then reused. An object pool supports the pooling of objects waiting to be reused. These object pools are not meant to be used for pooling Java Database Connectivity connections or Java Message Service (JMS) connections and sessions. WebSphere Application Server provides specialized mechanisms for dealing with those types of objects. These object pools are intended for pooling application-defined objects or basic Developer Kit types.

To view this administrative console page, click **Resources > Object pool managers >** *objectpoolmanager_name*

To use an object pool, the product administrator must define an object pool manager using the administrative console. Multiple object pool managers can be created in an Application Server cell.

Scope:

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Name:

The name by which the object pool manager is known for administrative purposes.

Data type	String
Range	1 through 30 ASCII characters

JNDI Name:

The Java Naming and Directory Interface (JNDI) name for the object pool manager.

Data type	String

Description:

A description of the object pool manager.

Data type	String

Category:

A category name used to classify or to group this object pool manager.

Data type	String

Custom object pool collection:

An object pool manages a pool of arbitrary objects and helps applications avoid creating new Java objects repeatedly. Most objects can be created once, used and then reused. An object pool supports the pooling of objects waiting to be reused. These object pools are not meant to be used for pooling Java Database Connectivity connections or Java Message Service (JMS) connections and sessions. WebSphere Application Server provides specialized mechanisms for dealing with those types of objects. These object pools are intended for pooling application-defined objects or basic Developer Kit types.

To view this administrative console page, click **Resources > Object pool managers >** *objectpoolmanager_name* **> Custom object pools**.

Use custom object pools to insert additional logic around the following mechanisms:

- Constructing an object pool (A list of properties can be set)
- Flushing the object pool
- Getting objects from the pool
- Returning objects from the pool

These features allow for actions such as, clearing the state of an object when returning it to the pool, configuring the state of an object when retrieving it from the pool, or configuring generic pools and sending instructions on how to behave using custom properties.

To use an object pool the product administrator must define an object pool manager using the administrative console. You can create multiple object pool managers in an Application Server cell.

Pool class name:

Specifies the fully qualified class name of the objects that are stored in the custom object pool.

Data type String

Pool implementation class name:

Specifies the fully qualified class name of the implementation class for the custom object pool.

Data type String

Custom object pool settings:

An object pool manages a pool of arbitrary objects and helps applications avoid creating new Java objects repeatedly. Most objects can be created once, used and then reused. An object pool supports the pooling of objects waiting to be reused. These object pools are not meant to be used for pooling Java Database Connectivity connections or Java Message Service (JMS) connections and sessions. WebSphere Application Server provides specialized mechanisms for dealing with those types of objects. These object pools are intended for pooling application-defined objects or basic Developer Kit types.

To view this administrative console page, click **Resources > Object pool managers >** *objectpoolmanager_name* **> Custom object pools >** *objectpool_name*.

Use custom object pools to insert additional logic around the following mechanisms:

- Constructing an object pool (A list of properties can be set)
- Flushing the object pool
- Getting objects from the pool
- Returning objects from the pool

These features allow for actions such as, clearing the state of an object when returning it to the pool, configuring the state of an object when retrieving it from the pool, or configuring generic pools and sending instructions on how to behave using custom properties.

To use an object pool, the product administrator must define an object pool manager using the administrative console. Multiple object pool managers can be created in an Application Server cell.

Pool Class Name:

The fully qualified class name of the objects that are stored in the object pool.

Data type String

Pool Impl Class Name:

The fully qualified class name of the CustomObjectPool implementation class for this object pool.

Data type String

Object pool service settings

Use this page to enable or disable the object pool service, which manages object pool resources used by the server.

To view this administrative console page, click **Servers > Server Types > WebSphere application servers >** *server_name* > Container Services > **Object pool service**.

Enable service at server startup

Specifies whether the server attempts to start the object pool service.

Default
Cleared

Range

Selected
When the application server starts, it attempts to start the object pool service automatically.

Cleared
The server does not try to start the object pool service. If object pool resources are used on this server, then the system administrator must start the object pool service manually or select this property, and then restart the server.

Object pools: Resources for learning

This topic provides links to find relevant supplemental information about object pools.

Use the following links to find relevant supplemental information about object pools. The information resides on IBM and non-IBM Internet sites, whose sponsors control the technical accuracy of the information.

These links are provided for convenience. Often, the information is not specific to the IBM WebSphere Application Server product, but is useful all or in part for understanding the product. When possible, links are provided to technical papers and Redbooks that supplement the broad coverage of the release documentation with in-depth examinations of particular product areas.

Furthermore, these links provide guidance on using object pools. Since object pooling is a general topic and the WebSphere Application Server product implementation is only one way to use it, you must understand when object pooling is necessary. These articles help you make that decision.

Programming model and decisions

- Build your own ObjectPool in Java to boost application speed
- Improve the robustness and performance of your ObjectPool
- Recycle broken objects in resource pools

MBeans for object pool managers and object pools

Legacy MBean names for object pool managers and object pools are deprecated. The legacy names are based on the object pool manager name (which is not required to be unique) rather than the object pool manager JNDI name.

About this task

For object pools, the legacy name is also lacking any identifier of the version of the pooled class. Additionally, object pool Performance Monitoring Instrumentation (PMI) statistics are aggregated for object pools with the same legacy object pool MBean name.

For example, if the object pool manager and pooled class are as follows:

```
object pool manager name:               My ObjectPool
object pool manager JNDI name:          op/MyObjectPool
pooled class name:                      java.util.ArrayList
hash code of java.util.ArrayList.class: 1111eb3f (hexadecimal)
```

the legacy object pool manager MBean name will be:

```
ObjectPoolManager_My ObjectPool
```

and the legacy object pool MBean name will be:

```
ObjectPool_My ObjectPool_java.util.ArrayList
```

Instead of using the deprecated legacy MBean names, use the MBean names that are based on the JNDI name of the object pool manager.

For the example above, the JNDI name-based object pool manager MBean name is:

```
ObjectPoolManager_op/MyObjectPool
```

and the JNDI name-based object pool MBean name is:

```
ObjectPool_op/MyObjectPool_java.util.ArrayList.class@1111eb3f
```

Formats for MBean names

Type	Name format
Deprecated legacy object pool manager MBean name:	ObjectPoolManager_[object pool manager name]
JNDI name-based object pool manager MBean name:	ObjectPoolManager_[object pool manager JNDI name]
Deprecated legacy object pool MBean name:	ObjectPool_[object pool manager name]_[pooled class name]
JNDI name-based object pool MBean name:	ObjectPool_[object pool manager JNDI name]_[pooled class name].class@[hexadecimal representation of the hash code of the pooled class' java.lang.Class reference]

In all of the above formats, characters that are not valid for MBean names are replaced with the '.' character.

Chapter 18. Administering Object Request Broker (ORB)

This page provides a starting point for finding information about the Object Request Broker (ORB). The product uses an ORB to manage communication between client applications and server applications as well as among product components. These Java Platform, Enterprise Edition (Java EE) standard services are relevant to the ORB: Remote Method Invocation/Internet Inter-ORB Protocol (RMI/IIOP) and Java Interface Definition Language (Java IDL).

The ORB provides a framework for clients to locate objects in the network and call operations on those objects as though the remote objects were located in the same running process as the client, providing location transparency.

Administering Object Request Brokers

Object Request Broker service settings

Use this page to configure the Java Object Request Broker (ORB) service.

To view this administrative console page:

- For an application server, click **Servers > Server Types > WebSphere application servers >** *server_name* **> Container services > ORB service**.
- For a deployment manager, click **System administration > Deployment manager > ORB service**.

Several settings are available for controlling internal Object Request Broker (ORB) processing. You can use these settings to improve application performance in the case of applications that contain enterprise beans. You can make changes to these settings for the default server or any application server that is configured in the administrative domain.

Request timeout

Specifies the number of seconds to wait before timing out on a request message.

If you use command-line scripting, the full name of this system property is com.ibm.CORBA.RequestTimeout.

Data type	int
Units	Seconds
Default	180
Range	0 - largest integer recognized by Java

Request retries count

Specifies the number of times that the ORB attempts to send a request if a server fails. Retrying sometimes enables recovery from transient network failures. This field is ignored for z/OS.

If you use command-line scripting, the full name of this system property is com.ibm.CORBA.requestRetriesCount.

Data type	int
Default	1
Range	1 to 10

Request retries delay

Specifies the number of milliseconds between request retries. This field is ignored for z/OS.

If you use command-line scripting, the full name of this system property is com.ibm.CORBA.requestRetriesDelay.

Data type	int
Units	Milliseconds
Default	0
Range	0 to 60,000

Connection cache maximum

Specifies the maximum number of entries that can occupy the ORB connection cache before the ORB starts to remove inactive connections from the cache. This field is ignored for z/OS.

It is possible that the number of active connections in the cache will temporarily exceed this threshold value. If necessary, the ORB will continue to add connections as long as resources are available.

For use in command-line scripting, the full name of this system property is com.ibm.CORBA.MaxOpenConnections.

Data type	Integer
Units	Connections
Default	240
Range	10 - largest integer recognized by Java

Connection cache minimum

Specifies the minimum number of entries in the ORB connection cache. This field is ignored for z/OS.

The ORB will not remove inactive connections when the number of entries is below this value.

For use in command-line scripting, the full name of this system property is com.ibm.CORBA.MinOpenConnections.

Data type	Integer
Units	Connections
Default	100
Range	Any integer that is at least 5 less than the value specified for the Connection cache maximum property.

ORB tracing

Enables the tracing of ORB General Inter-ORB Protocol (GIOP) messages.

This setting affects two system properties: com.ibm.CORBA.Debug and com.ibm.CORBA.CommTrace. If you set these properties through command-line scripting, you must set both properties to `true` to enable the tracing of GIOP messages.

Data type	Boolean
Default	Not enabled (false)

Locate request timeout

Specifies the number of seconds to wait before timing out on a LocateRequest message. This field is ignored for z/OS.

If you use command-line scripting, the full name of this system property is com.ibm.CORBA.LocateRequestTimeout.

Data type	int
Units	Seconds
Default	180
Range	0 to 300

Force tunneling

Controls how the client ORB attempts to use HTTP tunneling. This field is ignored for z/OS.

If you use command-line scripting, the full name of this system property is com.ibm.CORBA.ForceTunnel.

Data type	String
Default	`NEVER`
Range	Valid values are `ALWAYS`, `NEVER`, or `WHENREQUIRED`.

Considering the following information when choosing the valid value:

ALWAYS
Use HTTP tunneling immediately, without trying TCP connections first.

NEVER
Disable HTTP tunneling. If a TCP connection fails, a CORBA system exception (COMM_FAILURE) occurs.

WHENREQUIRED
Use HTTP tunneling if TCP connections fail.

Tunnel agent URL

Specifies the web address of the servlet to use in support of HTTP tunneling. This field is ignored on the z/OS platform.

This web address must be a proper format:

```
http://w3.mycorp.com:81/servlet/com.ibm.CORBA.services.IIOPTunnelServlet
```

For applets: `http://applethost:port/servlet/com.ibm.CORBA.services.IIOPTunnelServlet`.

This field is required if HTTP tunneling is set. If you use command-line scripting, the full name of this system property is com.ibm.CORBA.TunnelAgentURL.

Pass by reference

Specifies how the ORB passes parameters. If enabled, the ORB passes parameters by reference instead of by value, to avoid making an object copy. If you do not enable the pass by reference option, a copy of the parameter passes rather than the parameter object itself. This can be expensive because the ORB must first make a copy of each parameter object.

You can use this option only when the Enterprise JavaBeans (EJB) client and the EJB are on the same classloader. This requirement means that the EJB client and the EJB must be deployed in the same EAR file.

If the Enterprise JavaBeans (EJB) client and server are installed in the same instance or the product, and the client and server use remote interfaces, enabling the pass by reference option can improve performance up to 50%. The pass by reference option helps performance only where non-primitive object types are passed as parameters. Therefore, int and floats are always copied, regardless of the call model.

gotcha: Enable this property with caution because unexpected behavior can occur. If an object reference is modified by the callee, the caller's object is modified as well, since they are the same object.

If you use command-line scripting, the full name of this system property is com.ibm.CORBA.iiop.noLocalCopies.

Data type	Boolean
Default	Not enabled (false)

The use of this option for enterprise beans with remote interfaces violates Enterprise JavaBeans (EJB) Specification, Version 2.0 (see section 5.4). Object references passed to Enterprise JavaBeans (EJB) methods or to EJB home methods are not copied and can be subject to corruption.

Consider the following example:

```
Iterator iterator = collection.iterator();
MyPrimaryKey pk = new MyPrimaryKey();
while (iterator.hasNext()) {
   pk.id = (String) iterator.next();
   MyEJB myEJB = myEJBHome.findByPrimaryKey(pk);
}
```

In this example, a reference to the same MyPrimaryKey object passes into the product with a different ID value each time. Running this code with pass by reference enabled causes a problem within the application server because multiple enterprise beans are referencing the same MyPrimaryKey object. To avoid this problem, set the com.ibm.websphere.ejbcontainer.allowPrimaryKeyMutation system property to `true` when the pass by reference option is enabled. Setting the pass by reference option to `true` causes the EJB container to make a local copy of the PrimaryKey object. As a result, however, a small portion of the performance advantage of setting the pass by reference option is lost.

As a general rule, any application code that passes an object reference as a parameter to an enterprise bean method or to an EJB home method must be scrutinized to determine if passing that object reference results in loss of data integrity or in other problems.

After examining your code, you can enable the pass by reference option by setting the com.ibm.CORBA.iiop.noLocalCopies system property to `true`. You can also enable the pass by reference option in the administrative console. Click **Servers > Server Types > Application servers >** *server_name* **> Container services > ORB Service** and select **Pass by reference**.

Object Request Broker custom properties

There are several ways to configure an Object Request Broker (ORB). For example, you can use ORB custom property settings, or system property settings to configure an ORB, or you can provide objects during ORB initialization. If you use the following ORB custom properties to configure an ORB, remember that two types of default values exist for some of these properties: the Java SE Development Kit (JDK) default values and the WebSphere Application Server default values.

The JDK default is the value that the ORB uses for a property if the property is not specified in any way. The WebSphere Application Server default is the value that the WebSphere Application Server sets for a property in one of the following files:

- The orb.properties file when an application server is installed.
- The server.xml file when an application server is configured.

Because WebSphere Application Server explicitly sets its default value, if both a WebSphere Application Server and a JDK default value are defined for a property, the WebSphere Application Server default takes precedence over the JDK default.

For more information about the different ways to specify ORB properties and the precedence order, read the JDK Diagnostic Guide for the version of the JDK that you are using.

The orb.properties file, that is located in the *was_home*/properties directory, contains ORB custom properties that are initially set to the WebSphere Application Server default values during the product installation process. These values are passed to the ORB in a properties object and take precedence over Java virtual machine (JVM) arguments and other orb.properties files in either the *java_home*/lib or *user_home* directories.

You can use the administrative console to specify new values for these ORB custom properties. Any value that you specify takes precedence over any JDK or WebSphere Application Server default values for these properties, including JVM arguments. The ORB custom properties settings that you specify in the administrative console are stored in the server.xml system file and are passed to an ORB in a properties object whenever an ORB is initialized.

To use the administrative console to set ORB custom properties, click **Servers > Server Types > Application servers >** *server_name* **> Container services > ORB service > Custom properties**. You can then change the setting of one of the listed custom properties or click **New** to add a new property to the list. Then, click **Apply** to save your change. When you finish the changes, click **OK** and then click **Save** to save your changes.

To use the java command on a command line, use the -D option; for example:

```
java -Dcom.ibm.CORBA.propname1=value1 -Dcom.ibm.CORBA.propname2=value2 ... application name
```

To use the launchclient command on a command line, prefix the property with -CC; for example:

```
launchclient yourapp.ear -CCDcom.ibm.CORBA.propname1=value1 -CCDcom.ibm.CORBA.propname2=value2
   ... optional application arguments
```

The Custom properties page might already include Secure Sockets Layer (SSL) properties that were added during product installation. A list of the additional properties that are associated with the ORB service follows. Unless otherwise indicated, the default values that are provided in the descriptions of these properties are the JDK default values.

You can use the custom properties page to define the following properties for use by the ORB.

- “com.ibm.CORBA.BootstrapHost” on page 1038
- “com.ibm.CORBA.BootstrapPort” on page 1038
- “com.ibm.CORBA.ConnectTimeout” on page 1038
- “com.ibm.CORBA.ConnectionInterceptorName” on page 1039
- “com.ibm.CORBA.enableLocateRequest” on page 1039
- “com.ibm.CORBA.FragmentSize” on page 1039
- “com.ibm.CORBA.ListenerPort” on page 1039
- “com.ibm.CORBA.LocalHost” on page 1039
- “com.ibm.CORBA.numJNIReaders” on page 1040
- “com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.JNIReaderPoolImpl” on page 1040
- “com.ibm.CORBA.RasManager” on page 1040
- “com.ibm.CORBA.ServerSocketQueueDepth” on page 1041
- “com.ibm.CORBA.ShortExceptionDetails” on page 1041
- “com.ibm.CORBA.WSSSLClientSocketFactoryName” on page 1041
- “com.ibm.CORBA.WSSSLServerSocketFactoryName” on page 1041
- “com.ibm.websphere.orb.threadPoolTimeout” on page 1041
- “com.ibm.websphere.threadpool.strategy.implementation” on page 1042
- “com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.calcinterval” on page 1042
- “com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.lruinterval” on page 1042
- “com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.outqueues” on page 1042

- “com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.statsinterval” on page 1043
- “com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.workqueue” on page 1043
- “com.ibm.ws.orb.services.redirector.MaxOpenSocketsPerEndpoint” on page 1043
- “com.ibm.ws.orb.services.redirector.RequestTimeout” on page 1043
- “com.ibm.ws.orb.transport.SSLHandshakeTimeout” on page 1044
- “com.ibm.ws.orb.transport.useMultiHome” on page 1044
- “javax.rmi.CORBA.UtilClass” on page 1044

com.ibm.CORBA.BootstrapHost

Specifies the domain name service (DNS) host name or IP address of the machine on which initial server contact for this client resides.

depfeat: This setting is deprecated.

For a command-line or programmatic alternative, read the topic *Client-side programming tips for the Object Request Broker service*.

com.ibm.CORBA.BootstrapPort

Specifies the port that the ORB uses to bootstrap to the machine on which the initial server contact for this client listens.

depfeat: This setting is deprecated.

For a command line or programmatic alternative, read the topic *Client-side programming tips for the Object Request Broker service*.

Default	2809

com.ibm.CORBA.ConnectTimeout

The com.ibm.CORBA.ConnectTimeout property specifies the maximum time, in seconds, that the client ORB waits before timing out when attempting to establish an IIOP connection with a remote server ORB. Typically, client applications use this property. You can specify the property for each individual application server through the administrative console.

Client applications can specify the com.ibm.CORBA.ConnectTimeout property in one of two ways:
- By including it in the orb.properties file
- By using the -CCD option to set the property with the launchclient script. The following example specifies a maximum timeout value of 10 seconds:

```
launchclient clientapp.ear -CCDcom.ibm.com.CORBA.ConnectTimeout=10...
```

Begin with the default timeout value, but consider factors such as network congestion and application server load and capacity. Lower values provide better failover performance in the case of extended problems with the remote server, such as downtime. Higher values are better for slow network or remote server performance. However, exceptions can occur if the remote server does not have enough time to complete the subsequent request. A value of `0` means that the ORB relies on the timeout set by the operating system TCP/IP layer. For most operating systems, the timeout is set to 75 seconds.

Note: The default for the com.ibm.CORBA.ConnectTimeout property for Version 8 is 10. Before Version 8, the default is 0.

Valid Range	0-300
Default	10

com.ibm.CORBA.ConnectionInterceptorName

Specifies the connection interceptor class that is used to determine the type of outbound IIOP connection to use for a request, and if secure, the quality of protection characteristics associated with the request.

WebSphere Application Server default	com.ibm.ISecurityLocalObjectBaseL13Impl.SecurityConnectionInterceptor
JDK default	None

com.ibm.CORBA.enableLocateRequest

Specifies whether the ORB uses the locate request mechanism to find objects in a WebSphere Application Server cell. Use this property for performance tuning.

When this property is set to `true`, the ORB first sends a short message to the server to find the object that it needs to access. This first contact is called the *locate request*. If most of your initial method invocations are small, setting this property to `false` might improve performance because this setting change can reduce the GIOP traffic by as much as one-half. If most of your initial method invocations are large, you should set this property to `true`. When the property is set to `true`, the small locate request message is sent instead of the large locate request message. The large message is then sent to the target after the desired object is found.

WebSphere Application Server default	true
JDK default	false

com.ibm.CORBA.FragmentSize

Specifies the size of GIOP fragments that the ORB uses when it sends requests. If the total size of a request exceeds the set value, the ORB breaks the request into fragments, and sends each fragment separately until the entire request is sent. Set this property on the client side with a -D system property if you use a stand-alone Java application.

Adjust the value specified for the com.ibm.CORBA.FragmentSize property if the amount of data that is sent over IIOP in most GIOP requests exceeds 1 KB, or if thread dumps show that most client-side threads are waiting while sending or receiving data. Most messages should have few or no fragments.

If you want to instruct the ORB not to chunk any of the requests or replies it sends, set this property to `0`. However, setting the value to zero does not prevent the ORB from receiving GIOP fragments in requests or replies sent by another ORB.

Units	Bytes.
Default	1024
Range	From 64 to the largest value of a Java integer type that is divisible by 8

com.ibm.CORBA.ListenerPort

Specifies the port on which this server listens for incoming requests. This setting only applies for client-side ORBs.

Default	Next available system-assigned port number
Range	0 - 2147483647

com.ibm.CORBA.LocalHost

Specifies the host name or IP address of the system on which the application server or client application ORB is running.

For application servers, this property is automatically set to the host name of the ORB_LISTENER_ADDRESS endpoint. Any value specified by the user will be overwritten by the ORB_LISTENER_ADDRESS host name.

- If the ORB_LISTENER_ADDRESS host name is "*", then the property is set to the local host name by using the InetAddress.getLocalHost().getCanonicalHostName() method.
- For client applications, if no value is specified for this property, the ORB obtains a value at run time by calling the InetAddress.getLocalHost().getHostAddress() method.

Note: Do not set this property to the "localhost" string or "127.0.0.1" value as these values can result in unpredictable behavior for both clients and servers. Those values might impact callback behavior, the use of server IORs, and the creation of ORB connections.

com.ibm.CORBA.numJNIReaders

Specifies the number of JNI reader threads to be allocated in the JNI reader thread pool that is used by the ORB. Each thread can handle up to 1024 connections.

gotcha: Before you specify this property, verify that a JSSE provider is selected as the provider for the SSL repertoire that is associated with the port on which the ORB service listens for incoming requests. You can specify either `IBMJSSE2 SSL` or `IBMJSSE SSL`. `IBMJSSE2 SSL` is the default provider setting for SSL repertoires.

Valid Range	1 - 2147483647
Default	4

com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.JNIReaderPoolImpl

Specifies that JNI reader threads are used. The property name specifies the class name of the ORB component that manages the pool of JNI reader threads and interacts with the native OS library used to process multiple connections simultaneously.

gotcha:

- Verify that the library is located in the bin directory for the product.

  For an Intel operating system, the name of the file that contains the library name is `Selector.dll`

  For a UNIX-based operating system, the name of the file that contains the library is either `libSelector.a` or `libSelector.so`. If the lib prefix is missing from the file name, rename the file such that the name includes the lib prefix.
- When you specify this property using the administrative console, enter `com.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.JNIReaderPoolImpl` for the property name and an empty string ("") for the value.

  When you specify this property on the java command, do not include a value:

  `-Dcom.ibm.CORBA.ORBPluginClass.com.ibm.ws.orbimpl.transport.JNIReaderPoolImpl`

Valid Range	Not applicable
Default	None

com.ibm.CORBA.RasManager

Specifies an alternative to the default RAS manager of the ORB. This property must be set to com.ibm.websphere.ras.WsOrbRasManager before the ORB can be integrated with the rest of the RAS processing for the product.

WebSphere Application Server default	com.ibm.websphere.ras.WsOrbRasManager
JDK default	None

com.ibm.CORBA.ServerSocketQueueDepth

Specifies the maximum number of connection requests that can be waiting to be handled by the Server ORB before the product starts to reject new incoming connection requests. This property corresponds to the `backlog` argument to a ServerSocket constructor and is handled directly by TCP/IP.

If you see a "connection refused" message in a trace log, typically, either the port on the target machine is not open, or the server is overloaded with queued-up connection requests. Increasing the value specified for this property can help alleviate this problem if there does not appear to be any other problem in the system.

Default	50
Range	From 50 to the largest value of the Java int type

com.ibm.CORBA.ShortExceptionDetails

Specifies that the exception detail message that is returned whenever the server ORB encounters a CORBA system exception contains a short description of the exception as returned by the toString method of java.lang.Throwable class. Otherwise, the message contains the complete stack trace as returned by the printStackTrace method of java.lang.Throwable class.

com.ibm.CORBA.WSSSLClientSocketFactoryName

Specifies the class that the ORB uses to create SSL sockets for secure outbound IIOP connections.

WebSphere Application Server default	com.ibm.ws.security.orbssl.WSSSLClientSocketFactoryImpl
JDK default	None

com.ibm.CORBA.WSSSLServerSocketFactoryName

Specifies the class that the ORB uses to create SSL sockets for inbound IIOP connections.

WebSphere Application Server default	com.ibm.ws.security.orbssl.WSSSLServerSocketFactoryImpl
JDK default	None

com.ibm.websphere.orb.threadPoolTimeout

Use this custom property to specify the length of time in which the object request broker (ORB) waits for an available thread from the ORB thread pool before rejecting a request.

When the ORB receives an incoming request, the request is read by an ORB reader thread. The ORB reader thread attempts to hand off the request for processing by a worker thread in the ORB thread pool. When all the worker threads are handling other requests, the reader thread waits until a worker thread becomes available. While the reader thread is waiting, new requests are not processed by that particular reader thread. This situation can lead to deadlocks between the ORB thread pools on two different Java virtual machine (JVM) processes. The deadlocks are prominent when the ORB in one JVM process must call back to the ORB in the other JVM process to complete its request. Therefore, it is highly advisable to set this property to a positive non-zero value, which configures a finite wait period and can limit deadlock situations. However, configure the value for this custom property based on its effect on the average request processing time, the ORB request timeout value, and whether servers are making additional circular or backend calls.

Data type	Integer
Units	Milliseconds
Default	0
Range	0 - largest integer that is recognized by the Java run time

com.ibm.websphere.threadpool.strategy.implementation

Specifies the logical pool distribution (LPD) thread pool strategy that takes effect the next time you start the application server, and is enabled if set to com.ibm.ws.threadpool.strategy.LogicalPoolDistribution.

depfeat: The logical pool distribution function is deprecated. Do not configure logical pool distribution unless you have already configured it for a previous release of the product.

Some requests have shorter start times than others. LPD is a mechanism for providing these shorter requests more access to start threads. For more information, read the topic *Logical pool distribution*, that is included in the product information center.

com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.calcinterval

Specifies how often the logical pool distribution (LPD) mechanism readjusts the pool start target times. This property cannot be turned off after this support is installed.

depfeat: The logical pool distribution function is deprecated. Do not configure logical pool distribution unless you have already configured it with a previous release of the product.

If you use this property, LPD must be enabled. Read the description of the com.ibm.websphere.threadpool.strategy.implementation property for more information.

Data type	Integer
Units	Milliseconds
Default	30
Range	20,000 milliseconds minimum

com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.lruinterval

Specifies, in milliseconds, how long the logical pool distribution internal data is kept for inactive requests. This mechanism tracks several statistics for each request type that is received. Consider removing requests that have been inactive for an unusually long length of time.

depfeat: This function is deprecated. Do not configure logical pool distribution unless you have already configured it with a previous release of the product.

If you use this property, LPD must be enabled. Read the description of the com.ibm.websphere.threadpool.strategy.implementation property for more information.

Data type	Integer
Units	Milliseconds
Default	300000 (5 minutes)
Range	60000 (1 minute) minimum

com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.outqueues

Specifies how many pools are created and how many threads are allocated to each pool in the logical pool distribution mechanism.

depfeat: The logical pool distribution function is deprecated. Do not configure logical pool distribution unless you have already configured it with a previous release of the product.

The ORB parameter for specifying the maximum number of threads controls the total number of threads. The outqueues parameter is specified as a comma-separated list of percentages that add up to 100. For example, the list 25,25,25,25 sets up 4 pools, each allocated 25 percent of the available ORB thread pool. The pools are indexed left to right from 0 to n-1. The calculation mechanism dynamically assigns each outqueue a target start time. Target start times are assigned to outqueues in increasing order. Therefore, pool 0 gets the requests with the least start time, and pool n-1 gets requests with the highest start times.

If you specify this property, LPD must be enabled. Read the description of the com.ibm.websphere.threadpool.strategy.implementation property for more information.

Data type	Integers in comma-separated list
Default	25,25,25,25
Range	Percentages in list must total 100 percent

com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.statsinterval

Specifies that statistics are dumped to stdout after this interval expires, but only if requests are processed. This process keeps the mechanism from filling the log files with redundant information. These statistics are beneficial for tuning the logical pool distribution mechanism.

depfeat: The logical pool distribution function is deprecated. Do not configure logical pool distribution unless you have already configured it with a previous release of the product.

If you use this property, LPD must be enabled. Read the description of the com.ibm.websphere.threadpool.strategy.implementation property for more information.

Data type	Integer
Units	Milliseconds
Default	0 (off)
Range	30,000 (30 seconds) minimum

com.ibm.websphere.threadpool.strategy.LogicalPoolDistribution.workqueue

Specifies the size of a new queue where incoming requests wait for dispatch. Pertains to the logical pool distribution mechanism.

depfeat: The logical pool distribution function is deprecated. Do not configure logical pool distribution unless you have already configured it with a previous release of the product.

If you use this property, LPD must be enabled. Read the description of the com.ibm.websphere.threadpool.strategy.implementation property for more information.

Data type	Integer
Default	96
Range	10 minimum

com.ibm.ws.orb.services.redirector.MaxOpenSocketsPerEndpoint

Specifies the maximum number of connections that the IIOP Tunnel Servlet maintains in its connection cache for each target host and port. If the number of concurrent client requests to a single host and port exceeds the setting for this property, the IIOP Tunnel Servlet opens a temporary connection to the target server for each extra client request, and then closes the connection after it receives the reply. Connections that are opened, but not used within 5 minutes, are removed from the cache for the IIOP Tunnel Servlet.

WebSphere Application Server default	3
JDK default	Not applicable
Range	0 - largest integer recognized by Java

com.ibm.ws.orb.services.redirector.RequestTimeout

Specifies the number of seconds that the IIOP Tunnel Servlet waits for a reply from the target server on behalf of a client before timing out. If a value is not specified for this property, or is incorrectly specified, the com.ibm.CORBA.RequestTimeout property setting for the application server, on which the IIOP Tunnel Servlet is installed, is used as the setting for the com.ibm.ws.orb.services.redirector.RequestTimeout property.

The value you specify for this property must be at least as high as the highest client setting for the com.ibm.CORBA.RequestTimeout property; otherwise the IIOP Tunnel Servlet might timeout more quickly than the client typically times out while waiting for a reply. If this property is set to zero, the IIOP Tunnel Servlet does not timeout.

WebSphere Application Server default	com.ibm.CORBA.RequestTimeout property setting for the application server on which the IIOP Tunnel Servlet is installed. The request_timeout request-level Reliability Availability and Serviceability (RAS) attribute overrides the com.ibm.CORBA.RequestTimeout property for IIOP requests. You define the request-level RAS attributes in the workload classification file.
JDK default	Not applicable
Range	0 - largest integer recognized by Java

com.ibm.ws.orb.transport.SSLHandshakeTimeout

This custom property specifies a timeout value for reading Secure Sockets Layer (SSL) handshake-related messages.

When you set the com.ibm.ws.orb.transport.SSLHandshakeTimeout custom property to a positive integer value, the listener thread does not hang if a message is received after the specified timeout period. If you do not set this custom property or do not set it to a positive integer value, the timeout value defaults to zero (0), a timeout period is not added, and the listener thread can hang.

Data type	Integer
Default	Zero (0)
Units	Milliseconds

com.ibm.ws.orb.transport.useMultiHome

Specifies whether the server ORB binds to all network interfaces in the system. If you specify `true`, the ORB binds to all network interfaces that are available to it. If you specify `false`, the ORB only binds to the network interface that is specified for the com.ibm.CORBA.LocalHost system property.

WebSphere Application Server default	true
JDK default	true

javax.rmi.CORBA.UtilClass

Specifies the name of the Java class that the product uses to implement the javax.rmi.CORBA.UtilDelegate interface.

This property supports delegation for method implementations in the javax.rmi.CORBA.Util class. The javax.rmi.CORBA.Util class provides utility methods that can be used by stubs and ties to perform common operations. The delegate is a singleton instance of a class that implements this interface and provides a replacement implementation for all of the methods of javax.rmi.CORBA.Util. To enable a delegate, provide the class name of the delegate as the value of the javax.rmi.CORBA.UtilClass system property. The default value provides support for the com.ibm.CORBA.iiop.noLocalCopies property.

WebSphere Application Server default	com.ibm.ws.orb.WSUtilDelegateImpl
JDK default	None

Character code set conversion support for the Java Object Request Broker service

The CORBA/IIOP specification defines a framework for negotiation and conversion of character code sets used by the Java Object Request Broker (ORB) service.

This product supports the framework and provides the following system properties for modifying the default settings:

com.ibm.CORBA.ORBCharEncoding

Specifies the name of the native code set that the ORB uses for character data (referred to as *NCS-C* in the CORBA/IIOP specification). By default, the ORB uses `UTF8`. Valid code set values for this property are shown in the table that follows this list; values that are valid only for ORBWCharDefault are indicated.

com.ibm.CORBA.ORBWCharDefault

Specifies the default code set that the ORB uses for transmission of wide character data when no code set for wide character data is found in the tagged component in the Interoperable Object Reference (IOR) or in the GIOP service context. If no code set for wide character data is found and this property is not set, the ORB raises an exception, as specified in the CORBA specification. No default value is set for this property. The only valid code set values for this property are `UCS2` or `UTF16`.

The CORBA code set negotiation and conversion framework specifies the use of code set registry IDs as defined in the Open Software Foundation (OSF) code set registry. The ORB translates the Java file.encoding names shown in the following table to the corresponding OSF registry IDs. These IDs are then used by the ORB in the IOR Code set tagged component and GIOP code set service context as specified in the CORBA and IIOP specification.

Table 68. OSF code set registry IDs for Java file.encoding names. The following table lists the OSF code set registry IDs for the Java file.encoding names.

Java name	OSF registry ID	Comments
ASCII	0x00010020	
ISO8859_1	0x00010001	
ISO8859_2	0x00010002	
ISO8859_3	0x00010003	
ISO8859_4	0x00010004	
ISO8859_5	0x00010005	
ISO8859_6	0x00010006	
ISO8859_7	0x00010007	
ISO8859_8	0x00010008	
ISO8859_9	0x00010009	
ISO8859_15_FDIS	0x0001000F	
Cp1250	0x100204E2	
Cp1251	0x100204E3	
Cp1252	0x100204E4	
Cp1253	0x100204E5	
Cp1254	0x100204E6	
Cp1255	0x100204E7	
Cp1256	0x100204E8	
Cp1257	0x100204E9	

Table 68. OSF code set registry IDs for Java file.encoding names (continued). The following table lists the OSF code set registry IDs for the Java file.encoding names.

Java name	OSF registry ID	Comments
Cp943C	0x100203AF	
Cp943	0x100203AF	
Cp949C	0x100203B5	
Cp949	0x100203B5	
Cp1363C	0x10020553	
Cp1363	0x10020553	
Cp950	0x100203B6	
Cp1381	0x10020565	
Cp1386	0x1002056A	
EUC_JP	0x00030010	
EUC_KR	0x0004000A	
EUC_TW	0x00050010	
Cp964	0x100203C4	
Cp970	0x100203CA	
Cp1383	0x10020567	
Cp33722C	0x100283BA	
Cp33722	0x100283BA	
Cp930	0x100203A2	
Cp1047	0x10020417	
UCS2	0x00010100	Valid only for the ORBWCharDefault
UTF8	0x05010001	
UTF16	0x00010109	Valid only for the ORBWCharDefault

Chapter 19. Administering OSGi Applications

Configure an enterprise bundle archive (EBA) asset, and maintain the bundle versions used by the EBA asset. Administer bundles, composite bundles, composite bundle extensions, and bundle repositories.

Before you begin

This topic assumes that you have already packaged your OSGi application as an enterprise bundle archive (EBA) file, and imported it into WebSphere Application Server, as described in Developing and deploying an OSGi application. You might also have added one or more composite bundle extensions to the deployed application.

Procedure

- Check the bundle download status of an EBA asset.

  Updating an asset to use a new bundle version might require bundle downloads. You cannot update an asset until bundle downloads are complete from any previous update. Before you try and update bundle versions, you can check the bundle download status of the asset. This status is either “Bundles downloading...”, “Bundle downloads are complete”, or “No bundles downloads are required”.
- Check the update status of an OSGi composition unit.

  When you change the versions of bundles or composite bundles that an enterprise OSGi application uses, or add or remove a composite bundle extension, the bundles used by the deployed application can get out of synchronization with those that are available. To check that your business-level application is running the most recent version of the EBA asset and any composite bundle extensions, you check the update status of the associated OSGi composition unit.
- Update bundle versions for an EBA asset.

  After you import your OSGi application as an asset, newer versions of the bundles or composite bundles that the asset uses might become available. You can configure the deployed asset to use an updated version of any bundle or composite bundle that is used by the asset. You can choose to use a specific bundle version, or to pull in the latest compatible version. Use either of the following methods:
  - Update bundle versions using the administrative console.
  - Update bundle versions using the editAsset command.
- Maintain an OSGi composition unit.

  Whenever an enterprise bundle archive (EBA) asset is updated, you can (optionally) update the associated composition unit. You might also update a composition unit to add or remove a composite bundle extension. If any updates need configuration changes, you can also modify the configuration information for the composition unit.
- Administer bundle repositories.

  OSGi applications can share a large number of common utility bundles. To simplify maintenance, and reduce the application footprint, the application does not need to include its own copy of each utility bundle. Instead, bundles can be hosted in a bundle repository, from where they are retrieved during deployment. You can administer bundles and composite bundles held in the internal bundle repository of the product, and also add links to external bundle repositories.
- Export and import a deployment manifest file.

  You can export the deployment manifest file from an application, then import the manifest file into another instance of the same application located somewhere else. This process is useful when an application moves from one environment to another, for example from a test environment to a production environment.

Updating bundle versions for an EBA asset

After you import your OSGi application as an asset, newer versions of the bundles or composite bundles that the asset uses might become available. You can configure the deployed asset to use an updated version of any bundle or composite bundle that is used by the asset. You can choose to use a specific bundle version, or to pull in the latest compatible version.

Before you begin

You can update bundle and composite bundle versions for an EBA asset by using the administrative console as described in this topic, or by using wsadmin commands as described in “Updating bundle versions for an EBA asset using the editAsset command” on page 1049.

About this task

When you first create an OSGi application, each bundle and composite bundle in the application is either directly contained in the EBA file or pulled in by reference. After you import your application as an asset, you can no longer change the direct contents of the asset. To update bundles and composite bundles that are specified in an asset, you add the updated versions to a repository then apply the updates to the asset. The asset is not updated automatically when new bundle versions become available; it is up to you to decide if and when to update the asset.

For each bundle or composite bundle specified by your EBA asset you can select either of the following options:

- Use a specific available bundle version.
- Use any version. In this case, the latest available version that is compatible with your selections for other bundles is used.

After you make your selections for this asset, the system tries to resolve the changes you have requested, and shows you the results. When you have selected a working configuration, you can commit your selections and the bundle version updates for the asset are applied.

You cannot update bundles that are provisioned by the runtime environment.

Note: When an EBA asset is updated, the update does not automatically affect the running business-level application. To update the running application, you update the composition unit of the business-level application that contains the asset.

For transitioning users: In the WebSphere Application Server Version 7 Feature Pack for OSGi Applications and Java Persistence API 2.0, bundle changes to the asset are applied by restarting the business-level application, rather than by updating the composition unit. The current approach means that many bundle changes can be applied in place, without restarting the running business-level application.

Procedure

1. Start the administrative console.
2. Navigate to **Applications** > **Application Types** > **Assets** > ***asset_name***.

   The Asset settings panel is displayed.
3. Check the current bundle download status for all bundles and composite bundles for this asset.

   If the asset has previously been updated, the bundle downloads for the previous update must have completed.

   The current bundle download status for all bundles and composite bundles for this asset is displayed in the EBA Dependencies section. This status is either “Bundles downloading...”, “Bundle downloads are complete”, or “No bundles downloads are required”.

If the bundle downloads for any previous update have completed, the option to update bundle versions is available under the Additional Properties section.

4. Click **[Additional Properties] Update bundle versions in this application**. The Update bundle versions in this application [Settings] panel is displayed. This panel contains a tabular overview of the bundles and composite bundles currently deployed in the asset. *Application bundles* are listed separately from *use bundles*.
5. Choose the update bundle version preference for each bundle in this application.

   In the table, you can select the update preference for each bundle from a list. You can choose either of the following options:

   - Choose a specific available bundle version. For example, “1.0.0”.
   - Choose “No preference”. If you select this option, the latest available version that is compatible with your other bundle selections is used.
6. Click **Preview**. A similar table is displayed in the Preview bundle versions update [Settings] panel, showing the result of the proposed changes to the bundle versions in this application. If the changes resolve successfully, the following message is displayed: “The selected bundle versions can be resolved, so you can now create a new deployment with the proposed bundle versions. The new deployment will not affect any composition units for this asset until the composition units are updated to use the new deployment.” Otherwise, the message displayed is “The selected versions cannot be resolved, so you cannot create a new deployment with the proposed bundle versions.”
7. Optional: If the changes have not been successfully resolved, click **Cancel** to return to the Update bundle versions in this application [Settings] panel, then select a different update preference for each bundle that could not be resolved.
8. When you have selected a working configuration, click **Create**. The bundle and composite bundle version updates for the asset are applied.
9. Save your changes to the master configuration. The bundle updates are downloaded.

What to do next

If you plan to update the composition unit at this time, check that all bundle downloads are complete. See “Checking the update status of an OSGi composition unit” on page 1077.

Updating bundle versions for an EBA asset using the editAsset command

After you import your OSGi application as an asset, newer versions of the bundles or composite bundles that the asset uses might become available. You can use the editAsset command to configure the deployed asset to use an updated version of any bundle or composite bundle that is used by the asset. You can choose to use a specific bundle version, or to pull in the latest compatible version.

Before you begin

You can update bundle and composite bundle versions for an EBA asset by using wsadmin commands as described in this topic, or by using the administrative console as described in “Updating bundle versions for an EBA asset” on page 1048.

About this task

When you first create an OSGi application, each bundle and composite bundle in the application is either directly contained in the EBA file or pulled in by reference. After you import your application as an asset, you can no longer change the direct contents of the asset. To update bundles and composite bundles that are specified in an asset, you add the updated versions to a repository then apply the updates to the asset. The asset is not updated automatically when new bundle versions become available; it is up to you to decide if and when to update the asset.

For each bundle or composite bundle specified by your EBA asset you can select either of the following options:

- Use a specific available bundle version.
- Use any version. In this case, the latest available version that is compatible with your selections for other bundles is used.

After you make your selections for this asset, the system tries to resolve the changes you have requested, and shows you the results. When you have selected a working configuration, the bundle and composite bundle version updates for the asset are applied.

You cannot update bundles that are provisioned by the runtime environment.

Note: When an EBA asset is updated, the update does not automatically affect the running business-level application. To update the running application, you update the composition unit of the business-level application that contains the asset.

For transitioning users: In the WebSphere Application Server Version 7 Feature Pack for OSGi Applications and Java Persistence API 2.0, bundle changes to the asset are applied by restarting the business-level application, rather than by updating the composition unit. The current approach means that many bundle changes can be applied in place, without restarting the running business-level application.

Procedure

1. Check the current bundle download status for all bundles in this asset.

   If the asset has previously been updated, the bundle downloads for the previous update must have completed.

   You can use the editCompUnit wsadmin command to check the bundle download status for an asset. This command checks the status of the associated OSGi composition unit, as described in “Checking the update status of an OSGi composition unit” on page 1077. This status is one of the following values:

   - Using latest OSGi application deployment.
   - New OSGi application deployment not yet available because it requires bundles that are still downloading.
   - New OSGi application deployment available.
   - New OSGi application deployment cannot be applied because bundle downloads have failed.

   Wait until the bundle downloads for any previous update have completed.
2. Choose the update bundle version preference for each bundle in this application.

   To select bundle versions for an EBA asset using the editAsset command, open a wsadmin command prompt then run the following jython command. Under the **-UpdateAppContentVersions** parameter, include an entry (that is, the *bundle_name current_version* and *update_preference*) for each bundle that is listed in the application manifest between the application content header and the use bundle header. Include every bundle, whether or not you are updating the bundle version.

   For transitioning users: In the WebSphere Application Server Version 7 Feature Pack for OSGi Applications and Java Persistence API 2.0, bundle changes to the asset are applied by restarting the business-level application. In Version 8, these changes are applied by updating the composition unit. To enable the current approach, the **UpdateAppContentVersionsStep** parameter has been replaced with the **UpdateAppContentVersions** parameter, and instead of restarting the business-level application you run the editCompUnit command with the **CompUnitStatusStep** parameter. See the following troubleshooting tip: The behavior has changed for using wsadmin commands to update bundle versions.

```
AdminTask.editAsset('[
  -assetID asset_name
  -UpdateAppContentVersions [
    [bundle_1_name current_version update_preference]
    [bundle_2_name current_version update_preference]
    [bundle_3_name current_version update_preference]
    [bundle_4_name current_version update_preference]
    [bundle_5_name current_version update_preference]
  ]]')
```

Notes:

- *current_version* specifies either a bundle version number, for example `1.0.0`, or `NOT_DEPLOYED` for shared bundles (that is, *use bundles*) that are declared in the application manifest but not deployed by the runtime environment. This argument describes the current configuration of the bundle, and is not used to change the configuration.
- *update_preference* specifies the new bundle version preference. This is either a bundle version number, for example `1.0.0`, or `NOT_DEPLOYED` for shared bundles, or `NO_PREF` if you want the system to choose a bundle version for you. If you do not want to update the version for a given bundle, set this attribute to the same value that you are using for the *current_version* attribute.

For more information about using the editAsset command, see BLAManagement command group for the AdminTask object using wsadmin scripting.

After you make your selections for this asset, the system tries to resolve the changes you have requested, and shows you the results. When you have selected a working configuration, the bundle and composite bundle version updates for the asset are applied.

3. Save your changes to the master configuration.

   To save your configuration changes, use the following command:

   ```
   AdminConfig.save()
   ```

   The bundle updates are downloaded.

What to do next

If you plan to update the composition unit at this time, check that all bundle downloads are complete. See “Checking the update status of an OSGi composition unit” on page 1077.

Maintaining an OSGi composition unit

Whenever an enterprise bundle archive (EBA) asset is updated, you can (optionally) update the associated composition unit. You might also update a composition unit to add or remove a composite bundle extension. If any updates need configuration changes, you can also modify the configuration information for the composition unit.

About this task

An OSGi composition unit consists of an EBA asset, (optionally) one or more composite bundle extensions, and configuration information for running the asset and composite bundle extensions in a business-level application. If a new OSGi application deployment is available, you can update the OSGi composition unit so that the business-level application uses the newer configuration.

If any updates need configuration changes, you can also modify the configuration information for the composition unit. The configuration information can include HTTP session management, context roots, virtual hosts, security roles, run-as roles, and web application or Blueprint resource reference bindings for your OSGi application.

Procedure

- Add or remove composite bundle extensions for an OSGi composition unit

  Use either of the following methods:

  – Add or remove composite bundle extensions using the administrative console.
  – Add or remove composite bundle extensions using wsadmin commands.
- Update an OSGi composition unit.

  Use either of the following methods:

  – Update an OSGi composition unit using the administrative console.
  – Update an OSGi composition unit using the editCompUnit command.

  When all bundle downloads are complete, you can update the OSGi composition unit so that the business-level application uses the newer configuration. If any of the updates contain configuration options, you update the configuration information. You can also take the opportunity to make additional, non-essential configuration changes.
- Modify the configuration of an OSGi composition unit.

  You can modify the configuration information for an OSGi composition unit at any time by using either of the following methods:

  – Modify the configuration of an OSGi composition unit using the administrative console.
  – Modify the configuration of an OSGi composition unit using wsadmin commands.

Results

When you save the changes to the composition unit, the associated business-level application is updated to use the new configuration. If the business-level application is running, the bundle and configuration updates are applied immediately. If possible (that is, depending on the nature of the updates) the system applies the updates without restarting the application. If new packages are imported, or new services injected, then the application is restarted. These packages and services can come from newly-provisioned bundles, or from bundles that have already been provisioned.

Updating an OSGi composition unit

If a new OSGi application deployment is available, you can update the OSGi composition unit so that the business-level application uses the newer configuration. If any of the updates contain configuration options, a wizard prompts you to update the configuration information.

Before you begin

This topic assumes that you have either updated the asset that the composition unit contains, or added a composite bundle as an extension to the composition unit. You do not have to update the composition unit every time you update the asset or add a composite bundle extension. You cannot update the composition unit until all bundle downloads are complete.

You can update an OSGi composition unit by using the administrative console as described in this topic, or by using wsadmin commands as described in “Updating an OSGi composition unit using the editCompUnit command” on page 1054.

About this task

An OSGi composition unit consists of an EBA asset, (optionally) one or more composite bundle extensions, and configuration information for running the asset and composite bundle extensions in a business-level application. The configuration information can include HTTP session management, context roots, virtual hosts, security roles, run-as roles, and web application or Blueprint resource reference bindings for your OSGi application.

When all bundle downloads are complete, you can update the OSGi composition unit so that the business-level application uses the newer configuration. If any of the updates contain configuration options, a wizard prompts you to update the configuration information. You can also take the opportunity to make additional, non-essential configuration changes.

For transitioning users: In the WebSphere Application Server Version 7 Feature Pack for OSGi Applications and Java Persistence API 2.0, bundle changes to the asset are applied by restarting the business-level application, rather than by updating the composition unit. The current approach means that many bundle changes can be applied in place, without restarting the running business-level application.

This topic describes the specific task of updating an OSGi composition unit. The more generalized task of updating the configuration of any composition unit is described in Updating business-level applications.

Procedure

1. Start the administrative console.
2. Navigate to **Applications** > **Application Types** > **Business-level applications** > ***application_name*** > ***composition_unit_name***.

   The Composition unit settings panel is displayed. The deployment status is displayed under **[General Properties] OSGi application deployment status**, and shows one of the following values:

   - Using latest OSGi application deployment.
   - New OSGi application deployment not yet available because it requires bundles that are still downloading.
   - New OSGi application deployment available.
   - New OSGi application deployment cannot be applied because bundle downloads have failed.

   If the status is “New OSGi application deployment available”, the **Update to latest deployment ...** button is available.
3. Update the composition unit to use the latest version of the EBA asset or composite bundle extension.

   If the status is “New OSGi application deployment available”, click **Update to latest deployment ...**. The Preview composition unit upgrade [Settings] panel is displayed.

   Because multiple updates might be available, and because updates do not have to be applied immediately, you might not be fully aware of the changes that you are about to make to the deployed application. So that you can see the cumulative effect of all the changes, this panel displays the complete list, bundle by bundle, of the updates that are about to be applied. If the result is not what you want, you can cancel the update. Otherwise, click **OK**.
4. Update the configuration information for running the asset or composite bundle extension in the business-level application.

   Bundle changes might also require configuration changes to the composition unit. For example, if you update a bundle in an EBA asset, or replace a composite bundle extension, you might introduce a resource that needs additional configuration, such as a new or changed Blueprint resource reference, or security role mapping.

   If any of the updates contain configuration options, a wizard prompts you to update the configuration information. This wizard is based on the Set options settings wizard that you use when creating a new OSGi composition unit. See Adding an EBA asset to a composition unit using the administrative console.
5. Save your changes to the master configuration.

Results

When you save the changes to the composition unit, the associated business-level application is updated to use the new configuration. If the business-level application is running, the bundle and configuration updates are applied immediately. If possible (that is, depending on the nature of the updates) the system

applies the updates without restarting the application. If new packages are imported, or new services injected, then the application is restarted. These packages and services can come from newly-provisioned bundles, or from bundles that have already been provisioned.

Updating an OSGi composition unit using the editCompUnit command

Use the editCompUnit command to select a composition unit that contains an enterprise OSGi application, and update the composition unit so that the business-level application uses the newer configuration. If any of the updates contain configuration options, run the editCompUnit command a second time to update the configuration information. You can also take the opportunity to make additional, non-essential configuration changes.

Before you begin

This topic assumes that you have either updated the asset that the composition unit contains, or added a composite bundle as an extension to the composition unit. You do not have to update the composition unit every time you update the asset or add a composite bundle extension. You cannot update the composition unit until all bundle downloads are complete.

You can update an OSGi composition unit by using the editCompUnit command as described in this topic, or by using the administrative console as described in “Updating an OSGi composition unit” on page 1052.

About this task

An OSGi composition unit consists of an EBA asset, (optionally) one or more composite bundle extensions, and configuration information for running the asset and composite bundle extensions in a business-level application. The configuration information can include HTTP session management, context roots, virtual hosts, security roles, run-as roles, and web application or Blueprint resource reference bindings for your OSGi application.

When all bundle downloads are complete, you can update the OSGi composition unit so that the business-level application uses the newer configuration. If any of the updates contain configuration options, run the editCompUnit command a second time to update the configuration information. You can also take the opportunity to make additional, non-essential configuration changes.

For transitioning users: In the WebSphere Application Server Version 7 Feature Pack for OSGi Applications and Java Persistence API 2.0, bundle changes to the asset are applied by restarting the business-level application, rather than by updating the composition unit. The current approach means that many bundle changes can be applied in place, without restarting the running business-level application. See the following troubleshooting tip: The behavior has changed for using wsadmin commands to update bundle versions.

Procedure

1. Check the update status of the composition unit.

   There are four distinct deployment statuses for an OSGi composition unit:

   - Using latest OSGi application deployment.
   - New OSGi application deployment not yet available because it requires bundles that are still downloading.
   - New OSGi application deployment available.
   - New OSGi application deployment cannot be applied because bundle downloads have failed.

2. Update the composition unit to use the latest version of the EBA asset or composite bundle extension.

   If the status is “New OSGi application deployment available”, open a wsadmin command prompt then run the following jython command:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -CompUnitStatusStep [[asset_name.eba true]]
]')
```

For example:

```
AdminTask.editCompUnit('[
  -blaId WebSphere:blaname=test.app
  -cuId WebSphere:cuname=com.ibm.ws.eba.extension.componenttest_0001.eba
  -CompUnitStatusStep [[com.ibm.ws.eba.extension.componenttest.eba true]]
]')
```

3. If required, update the configuration information for running the asset in the business-level application.

   Bundle changes might also require configuration changes to the composition unit. For example, if you update a bundle in an EBA asset, or replace a composite bundle extension, you might introduce a resource that needs additional configuration, such as a new or changed Blueprint resource reference, or security role mapping.

   If any of the updates contain configuration options, run the editCompUnit command a second time to update the configuration information. You can also take the opportunity to make additional, non-essential configuration changes. See “Modifying the configuration of an OSGi composition unit using wsadmin commands” on page 1065.

4. Save your changes to the master configuration.

   To save your configuration changes, use the following command:

   ```
   AdminConfig.save()
   ```

Results

When you save the changes to the composition unit, the associated business-level application is updated to use the new configuration. If the business-level application is running, the bundle and configuration updates are applied immediately. If possible (that is, depending on the nature of the updates) the system applies the updates without restarting the application. If new packages are imported, or new services injected, then the application is restarted. These packages and services can come from newly-provisioned bundles, or from bundles that have already been provisioned.

Adding or removing extensions for an OSGi composition unit

Add composite bundle extensions to, or remove them from, a composition unit that contains an enterprise OSGi application.

Before you begin

This topic assumes that you have developed a composite bundle, and added it to the internal bundle repository or to an external repository that can process composite bundles. See Developing a composite bundle.

You can manage extensions for a composition unit by using the administrative console as described in this topic, or by using wsadmin commands as described in “Adding or removing extensions for an OSGi composition unit using wsadmin commands” on page 1056.

About this task

After you import the enterprise bundle archive (EBA) file for your OSGi application as an asset, you can update versions of existing bundles but you cannot add extra bundles to the asset. However, after you have added the asset as a composition unit to a business-level application, you can extend the business-level application by adding one or more composite bundles to the composition unit.

You can add a composite bundle to, or remove it from, a composition unit. To update an extension to one with newer constituent bundles, you remove the composite bundle from the composition unit then add a new version of the composite bundle.

You can also view read-only information about each listed composite bundle.

Procedure

1. Start the administrative console.
2. Navigate to **Applications** > **Application Types** > **Business-level applications** > ***application_name*** > ***composition_unit_name*** > **[Additional Properties] Manage extensions for this composition unit**.

   The Manage extensions for this composition unit [Collection] panel is displayed. This panel lists all the composite bundle extensions that are currently added to this composition unit. To view read-only information about a composite bundle, click the composite bundle name in the list.
3. Add or remove composite bundle extensions.
   - Add one or more composite bundles as extensions to a composition unit.
     a. In the Manage extensions for this composition unit [Collection] panel, click **Add**. The Add extensions [Collection] panel is displayed.

        This panel lists all the composite bundles that are currently available to be added to the composition unit. That is, all composite bundles that are installed in an available bundle repository, and that have not already been added as a composite bundle extension or used by a deployed EBA asset. To view read-only information about a composite bundle, click the composite bundle name in the list.
     b. Select one or more composite bundles to add.
     c. Click **Add**. The composite bundle is added, and displayed in the list of composite bundle extensions in the Manage extensions for this composition unit [Collection] form.
   - Remove one or more composite bundle extensions from a composition unit.
     a. In the Manage extensions for this composition unit [Collection] panel, select one or more composite bundle extensions to remove.
     b. Click **Remove**. The selected composite bundles are removed, and the updated list of composite bundle extensions is displayed.
4. Save your changes to the master configuration. If you extended a deployed OSGi application, the composite bundle, including its constituent bundles, is downloaded.

What to do next

If you plan to update the composition unit at this time, check that all bundle downloads are complete. See “Checking the update status of an OSGi composition unit” on page 1077.

Adding or removing extensions for an OSGi composition unit using wsadmin commands

Use the addOSGiExtension or addOSGiExtensions command to add composite bundle extensions to a composition unit that contains an enterprise OSGi application. Similarly, use the removeOSGiExtension or removeOSGiExtensions command to remove composite bundle extensions. Use the listOSGiExtensions command to list all the extensions that are currently added to the composition unit.

Before you begin

This topic assumes that you have developed a composite bundle, and added it to the internal bundle repository or to an external repository that can process composite bundles. See Developing a composite bundle.

You can manage extensions for a composition unit by using wsadmin commands as described in this topic, or by using the administrative console as described in “Adding or removing extensions for an OSGi composition unit” on page 1055.

About this task

After you import the enterprise bundle archive (EBA) file for your OSGi application as an asset, you can update versions of existing bundles but you cannot add extra bundles to the asset. However, after you have added the asset as a composition unit to a business-level application, you can extend the business-level application by adding one or more composite bundles to the composition unit.

You can add a composite bundle to, or remove it from, a composition unit. To update an extension to one with newer constituent bundles, you remove the composite bundle from the composition unit then add a new version of the composite bundle.

Procedure

1. Optional: List composite bundle extensions.

   Use the listOSGiExtensions command to list the symbolic names and versions of all the extensions that are currently added to a composition unit.

   ```
   AdminTask.listOSGiExtensions('-cuName cu_name')
   ```

   Note: The output from the listOSGiExtensions command is formatted so that you can copy the list of extensions, then paste them into the removeOSGiExtensions command.

   For more information, see “listOSGiExtensions command” on page 1061.

2. Add or remove composite bundle extensions.

   - Add one or more composite bundles as extensions to a composition unit.

     Use the addOSGiExtension or addOSGiExtensions command:

     ```
     AdminTask.addOSGiExtension('
       -cuName cu_name
       -symbolicName cba_symbolic_name
       -version cba_version
     ')
     ```

     ```
     AdminTask.addOSGiExtensions([
       '-cuName', 'cu_name',
       '-extensions',
         'cba1_symbolic_name;cba1_version
          cba2_symbolic_name;cba2_version
          cba3_symbolic_name;cba3_version
         '
     ])
     ```

     The composite bundle must be available in the internal bundle repository, or in an external repository that can process composite bundles.

     For more information, see “addOSGiExtension command” on page 1058 or “addOSGiExtensions command” on page 1060.

   - Remove one or more composite bundle extensions from a composition unit.

     Use the removeOSGiExtension or removeOSGiExtensions command:

     ```
     AdminTask.removeOSGiExtension('
       -cuName cu_name
       -symbolicName cba_symbolic_name
       -version cba_version
     ')
     ```

     ```
     AdminTask.removeOSGiExtensions([
       '-cuName', 'cu_name',
       '-extensions',
         'cba1_symbolic_name;cba1_version
     ```

```
        cba2_symbolic_name;cba2_version
        cba3_symbolic_name;cba3_version
      '
])
```

Note: The output from the listOSGiExtensions command is formatted so that you can copy the list of extensions, then paste them into the removeOSGiExtensions command.

For more information, see “removeOSGiExtension command” on page 1059 or “removeOSGiExtensions command” on page 1062.

3. Save your changes to the master configuration.

   To save your configuration changes, use the following command:

   ```
   AdminConfig.save()
   ```

   If you extended a deployed OSGi application, the composite bundle, including its constituent bundles, is downloaded.

What to do next

If you plan to update the composition unit at this time, check that all bundle downloads are complete. See “Checking the update status of an OSGi composition unit” on page 1077.

addOSGiExtension command:

Use the addOSGiExtension command to add a composite bundle as an extension to a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a composite bundle as an extension to a composition unit. The composite bundle must be available in the internal bundle repository, or in an external repository that can process composite bundles.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-symbolicName *cba_symbolic_name*
The non-localizable name for this composite bundle.

-version *cba_version*
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`. The symbolic name, together with the version, identifies a unique composite bundle.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.addOSGiExtension('
  -cuName cu_name
  -symbolicName cba_symbolic_name
  -version cba_version
')
```

removeOSGiExtension command:

Use the removeOSGiExtension command to remove a composite bundle extension from a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes a composite bundle extension from a composition unit.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-symbolicName *cba_symbolic_name*
The non-localizable name for this composite bundle.

-version *cba_version*
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`. The symbolic name, together with the version, identifies a unique composite bundle.

Note: You can use the listOSGiExtensions command to list the symbolic names and versions of all the extensions that are currently added to a composition unit.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeOSGiExtension('
  -cuName cu_name
  -symbolicName cba_symbolic_name
  -version cba_version
')
```

addOSGiExtensions command:

Use the addOSGiExtensions command to add several composite bundles as extensions to a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds several composite bundles as extensions to a composition unit. The composite bundles must be available in the internal bundle repository, or in an external repository that can process composite bundles.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-extensions

A list of the composite bundle extensions to be added. Each list entry contains the symbolic name and the version for a composite bundle. The symbolic name, together with the version, identifies a unique composite bundle.

cba_symbolic_name
The non-localizable name for this composite bundle.

cba_version
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.addOSGiExtensions([
  '-cuName', 'cu_name',
  '-extensions',
    'cba1_symbolic_name;cba1_version
     cba2_symbolic_name;cba2_version
     cba3_symbolic_name;cba3_version
    '
])
```

listOSGiExtensions command:

Use the listOSGiExtensions command to list the symbolic names and versions of all the extensions that are currently added to a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

This command lists the symbolic names and versions of all the extensions that are currently added to a composition unit. The output from the listOSGiExtensions command is formatted so that you can copy the list of extensions, then paste them into the removeOSGiExtensions command.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listOSGiExtensions('-cuName cu_name')
```

removeOSGiExtensions command:

Use the removeOSGiExtensions command to remove several composite bundle extensions from a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes several composite bundle extensions from a composition unit.

You can use the listOSGiExtensions command to list the symbolic names and versions of all the extensions that are currently added to a composition unit. The output from the listOSGiExtensions command is formatted so that you can copy the list of extensions, then paste them into the removeOSGiExtensions command.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-extensions

A list of the composite bundle extensions to be removed. Each list entry contains the symbolic name and the version for a composite bundle. The symbolic name, together with the version, identifies a unique composite bundle.

cba_symbolic_name
The non-localizable name for this composite bundle.

cba_version
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeOSGiExtensions([
  '-cuName', 'cu_name',
  '-extensions',
    'cba1_symbolic_name;cba1_version
     cba2_symbolic_name;cba2_version
     cba3_symbolic_name;cba3_version
    '
])
```

Modifying the configuration of an OSGi composition unit

You can modify the configuration information for a composition unit that contains an enterprise OSGi application. An OSGi composition unit consists of an EBA asset, (optionally) one or more composite bundle extensions, and configuration information for running the asset and composite bundle extensions in a business-level application. The configuration information can include HTTP session management, context roots, virtual hosts, security roles, run-as roles, and web application or Blueprint resource reference bindings for your OSGi application.

Before you begin

You can modify the configuration of an OSGi composition unit by using the administrative console as described in this topic, or by using wsadmin commands as described in “Modifying the configuration of an OSGi composition unit using wsadmin commands” on page 1065.

About this task

An OSGi composition unit consists of an EBA asset, (optionally) one or more composite bundle extensions, and some or all of the following configuration information:

- Mappings from the composition unit to a target application server, web server or cluster.
- Configuration of the application's session manager, context roots or virtual hosts.
- Bindings to any associated web applications or blueprint resource references.
- Mappings from security roles to particular users or groups.

You first specify the configuration of an EBA asset or composite bundle extension when you add it to a composition unit. If bundles in the asset or composite bundle extension are subsequently changed, or if you need to remap resources, you can update the configuration. For example, if you update a bundle in an EBA asset, or replace a composite bundle extension, you might introduce a resource that needs additional configuration, such as a new or changed Blueprint resource reference, or security role mapping.

This topic describes the specific task of modifying the configuration of an OSGi composition unit. The more generalized task of modifying the configuration of any composition unit is described in Updating business-level applications.

Procedure

1. Start the administrative console.
2. Navigate to **Applications** > **Application Types** > **Business-level applications** > ***application_name*** > ***composition_unit_name***.

   The Composition unit settings panel is displayed.
3. Under the **Additional Properties** section, click any of the following options then modify the settings as required.
   - **Session management**

     Configure HTTP session management for the web application bundles. For more information, see Configuring session management by level.

     The Session management option is only visible if the OSGi application uses web application bundles.
   - **Map context roots**

     Select a web application bundle (WAB) from the list, then enter the context root for the WAB. For example, `/sample`. For more information, see Context root for web application bundles [Settings].

     The Map context roots option is only visible if the application uses web application bundles.
   - **Virtual hosts**

     The list of available WABs in this asset is displayed. For each WAB, you can change the associated virtual host by selecting a different one from the list. If you specify an existing virtual host in the `ibm-web-bnd.xml` or `.xmi` file for a given WAB, the specified virtual host is set by default. Otherwise, the default virtual host setting is `default_host`. For more information, see Virtual hosts for web application bundles [Settings].

     The Virtual hosts option is only visible if the application uses web application bundles.
   - **Security role to user/group mapping**

     Change the security mapping as needed. For more information, see Security role to user or group mapping [Settings].

The Security role to user/group mapping option is only visible if the application uses security roles.

- **Map RunAs roles to users**

  You can map a specified user identity and password to a RunAs role. This enables you to specify application-specific privileges for individual users, so that they can run specific tasks using another user identity. For more information, see Map RunAs roles to users [Collection].

  The Map RunAs roles to users option is only visible if the application uses RunAs roles.

- **Bind Blueprint resource references**

  The list of available Blueprint resource references in this asset is displayed. For each reference, you can optionally select an authentication alias from the list. Default authentication aliases (from `ibm-eba-bnd.xml` files) are offered only if they exist on every target server or cluster. For more information, see Blueprint resource reference bindings [Settings].

  The Bind Blueprint resource references option is only visible if the bundle includes Blueprint resource reference declarations.

- **Web module message destination and resource environment reference bindings**

  The list of available web application message destination and resource environment references in this asset is displayed. That is, resources of type message-destination-ref (message destination reference) or resource-env-ref (resource environment reference), as defined in the Java specification JSR-250: Common Annotations for the Java Platform. For each reference, specify the JNDI name under which the resource is known in the runtime environment. For more information, see Web module message destination and resource environment reference bindings [Settings].

  The Web module message destination and resource environment reference bindings option is only visible if the bundle includes a web application.

- **Web module resource reference bindings**

  The list of available web application resource references in this asset is displayed. That is, resources of type resource-ref (resource reference), as defined in the Java specification JSR-250: Common Annotations for the Java Platform. For each reference, specify the JNDI name under which the resource is known in the runtime environment. Optionally, set authentication properties and extended data source custom properties, which affect how the resource is accessed at run time. To specify the JNDI name mapping, either type the JNDI name into the box, or click **Browse...** then select the resource reference from the list of available resources. To modify the authentication method or set extended data source custom properties, select a single reference then click **Modify Resource Authentication Method...** or **Extended Properties...**. For more information, see Web module resource reference bindings [Settings].

  The Web module resource reference bindings option is only visible if the bundle includes a web application.

4. Save your changes to the master configuration.

Results

When you save the changes to the composition unit, the associated business-level application is updated to use the new configuration. If the business-level application is running, the bundle and configuration updates are applied immediately. If possible (that is, depending on the nature of the updates) the system applies the updates without restarting the application. If new packages are imported, or new services injected, then the application is restarted. These packages and services can come from newly-provisioned bundles, or from bundles that have already been provisioned.

Modifying the configuration of an OSGi composition unit using wsadmin commands

You can use the editCompUnit command and the AdminConfig commands to modify the configuration information for a composition unit that contains an enterprise OSGi application. An OSGi composition unit consists of an EBA asset, (optionally) one or more composite bundle extensions, and configuration information for running the asset and composite bundle extensions in a business-level application. The

configuration information can include HTTP session management, context roots, virtual hosts, security roles, run-as roles, and web application or Blueprint resource reference bindings for your OSGi application.

Before you begin

You can modify the configuration of an OSGi composition unit by using wsadmin commands as described in this topic, or by using the administrative console as described in “Modifying the configuration of an OSGi composition unit” on page 1063.

About this task

An OSGi composition unit consists of an EBA asset, (optionally) one or more composite bundle extensions, and some or all of the following configuration information:

- Mappings from the composition unit to a target application server, web server or cluster.
- Configuration of the application's session manager, context roots or virtual hosts.
- Bindings to any associated web applications or blueprint resource references.
- Mappings from security roles to particular users or groups.

You first specify the configuration of an EBA asset or composite bundle extension when you add it to a composition unit. If bundles in the asset or composite bundle extension are subsequently changed, or if you need to remap resources, you can update the configuration. For example, if you update a bundle in an EBA asset, or replace a composite bundle extension, you might introduce a resource that needs additional configuration, such as a new or changed Blueprint resource reference, or security role mapping.

To configure all elements of the composition unit except the HTTP session manager, you use the editCompUnit command. To configure the HTTP session manager, you use the AdminConfig commands to configure the deployed object represented by the *appDeploy* variable.

In the following procedure, all the steps and substeps are optional. You only need to re-configure the elements that have changed.

Procedure

- Configure all elements of the composition unit except the HTTP session manager.

  Each of the following substeps describes the syntax for modifying a single element of the composition unit using the editCompUnit command. You can run the command once for each element, or you can modify several elements in a batch.

  For several of the elements, the values you specify include bundle identifiers. If your EBA asset includes or references composite bundles, the command syntax is slightly different. For clarity, the differences for composite bundles are described, step by step, in a linked topic.

  1. Identify the composition unit that you want to edit.

     You identify a particular composition unit by specifying its business-level application ID and composition unit ID. Whenever you run the editCompUnit command, you must always include these two parameters in the command.

     -blaID
     Specifies the configuration ID of the business-level application.

     -cuID
     Specifies the ID of the composition unit.

     The jython syntax for this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  ...
]')
```

For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  ...
]')
```

2. Map the target node and server, or the target cluster.

   You cannot edit the deployable unit URI (which, for a composition unit that contains an OSGi application, is `ebaDeploymentUnit`). You can edit the target node and server, or the target cluster. To add an additional target, you use the plus sign character (+) as a prefix.

   If the target is one cluster, the jython syntax for this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -MapTargets [[ebaDeploymentUnit WebSphere:cluster=cluster_name]]
]')
```

   For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -MapTargets [[ebaDeploymentUnit WebSphere:cluster=cluster1]]
]')
```

   If the target is two servers, the jython syntax for this aspect of the command is as follows::

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -MapTargets [
    [ebaDeploymentUnit WebSphere:node=node_name,server=server_name+
                       WebSphere:node=node2_name,server=server2_name]]
]')
```

   For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -MapTargets [
    [ebaDeploymentUnit WebSphere:node=node01,server=server1+
                       WebSphere:node=node01,server=web1]]
]')
```

3. Define WAB context roots.

   Context roots determine where the web pages of a particular web application bundle (WAB) are found at run time. The context root that you specify here is combined with the defined server mapping to compose the full URL that you enter to access the pages of the WAB. For example, if the application server default host is www.example.com:8080 and the context root of the WAB is /sample, the web pages are available at www.example.com:8080/sample.

   The jython syntax for this aspect of the command is as follows. The bundles listed under the **ContextRootStep** need to be all the WABs contained in the OSGi application.

   Note: For composite bundles, the syntax is slightly different. See Step: Define context roots for web application bundles (WABs) in composite bundles.

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -ContextRootStep [
    [bundle_symbolic_name_1 bundle_version_1 context_root_1]
    [bundle_symbolic_name_2 bundle_version_2 context_root_2]]
]')
```

For example, for an EBA file that contains two WABs (com.ibm.ws.eba.helloWorldService.web at version 1.0.0, which is to be mapped to /hello/web, and com.ibm.ws.eba.helloWorldService.withContextRoot at version 0.9.0, which is to be mapped to /hello/service), this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -ContextRootStep [
    [com.ibm.ws.eba.helloWorldService.web 1.0.0 "/hello/web"]
    [com.ibm.ws.eba.helloWorldService.withContextRoot 0.9.0 "/hello/service"]]
]')
```

4. Map WABs to virtual hosts.

   You use a virtual host to associate a unique port with a web application. The aliases of a virtual host identify the port numbers defined for that virtual host. A port number specified in a virtual host alias is used in the URL that is used to access artifacts such as servlets and JavaServer Page (JSP) files in a web application. For example, the alias myhost:8080 is the *host_name:port_number* portion of the URL http://myhost:8080/sample.

   Each WAB that is contained in a deployed asset must be mapped to a virtual host. WABs can be installed on the same virtual host, or dispersed among several virtual hosts.

   If you specify an existing virtual host in the ibm-web-bnd.xml or .xmi file for a given WAB, the specified virtual host is set by default. Otherwise, the default virtual host setting is default_host, which provides several port numbers through its aliases:

   80 An internal, insecure port used when no port number is specified
   9080 An internal port
   9443 An external, secure port

   Unless you want to isolate your WAB from other WABs or resources on the same node, default_host is a suitable virtual host. In addition to default_host, WebSphere Application Server provides admin_host, which is the virtual host for the administrative console system application. admin_host is on port 9060. Its secure port is 9043. Do not select admin_host unless the WAB relates to system administration.

   The jython syntax for this aspect of the command is as follows.

   Note: For composite bundles, the syntax is slightly different. See Step: Map WABs in composite bundles to virtual hosts.

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -VirtualHostMappingStep [
    [bundle_symbolic_name_1 bundle_version_1
    web_module_name_1 virtual_host_1]
    [bundle_symbolic_name_2 bundle_version_2
     web_module_name_2 virtual_host_2]]
]')
```

   For example, for an EBA file containing two WABs (com.ibm.ws.eba.helloWorldService.web at version 1.0.0, which is to be mapped to default_host, and com.ibm.ws.eba.helloWorldService.withContextRoot at version 0.9.0, which is to be mapped to test_host), this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -VirtualHostMappingStep [
    [com.ibm.ws.eba.helloWorldService.web 1.0.0
    "HelloWorld service" default_host]
    [com.ibm.ws.eba.helloWorldService.withContextRoot 0.9.0
    "HelloWorld second service" test_host]]
]')
```

5. Map security roles to users or groups.

   The jython syntax for this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -MapRolesToUsersStep [
    [role_name everyone?
    all_authenticated_in_realm?
    usernames groups]]
]')
```

   Key:

   - *role_name* is a role name defined in the application.
   - *everyone?* is set to Yes or No, to specify whether or not everyone is in the role.
   - *all_authenticated_in_realm?* is set to Yes or No, to specify whether or not all authenticated users can access the application realm.
   - *usernames* is a list of WebSphere Application Server user names, separated by the "|" character.
   - *groups* is a list of WebSphere Application Server groups, separated by the "|" character.

   Note: For *usernames*, and *groups*, the empty string "" means "use the default or existing value". The default value is usually that no users or groups are bound to the role. However, when an application contains an `ibm-application-bnd.xmi` file, the default value for *usernames* is obtained from this file. If you are deploying an application that contains an `ibm-application-bnd.xmi` file, and you want to remove the bound users, specify just the "|" character (which is the separator for multiple user names). This explicitly specifies "no users", and therefore guarantees that no users are bound to the role.

   For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -MapRolesToUsersStep [
    [ROLE1 No Yes "" ""]
    [ROLE2 No No WABTestUser1 ""]
    [ROLE3 No No "" WABTestGroup1]
    [ROLE4 Yes No "" ""]]
]')
```

   For more information about the -MapRolesToUsersStep option, see the information for the $AdminApp install command “MapRolesToUsers” option. This is the equivalent option for Java EE applications. For more general information, see Security role to user or group mapping.

6. Map RunAs roles to users

   You can map a specified user identity and password to a RunAs role. This enables you to specify application-specific privileges for individual users, so that they can run specific tasks using another user identity. The jython syntax for this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -MapRunAsRolesToUsersStep [
    [role_name user_name password]]
]')
```

   For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
```

```
  -MapRunAsRolesToUsersStep [
    [Role1 User1 password1]
    [AdminRole User3 password3]]
]')
```

For more information about the -MapRunAsRolesToUsers option, see the information for the $AdminApp install command “MapRunAsRolesToUsers” option. This is the equivalent option for Java EE applications. For more general information, see Map RunAs roles to users.

7. Add authentication aliases to Blueprint resource references.

   Blueprint components can access WebSphere Application Server resource references. Each reference is declared in a Blueprint XML file, and can be secured using a Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) authentication alias. Each bundle in an OSGi application can contain any number of resource reference declarations in its various Blueprint XML files.

   When you secure resource references, those resource references can be bound only to JCA authentication aliases that exist on every server or cluster on which the application is deployed. An OSGi application can be deployed to multiple servers and clusters that are in the same security domain. Therefore, each JCA authentication alias must exist in either the security domain of the target servers and clusters, or the global security domain.

   You must declare the resource references in the Blueprint XML file. For example:

```
<blueprint xmlns="http://www.osgi.org/xmlns/blueprint/v1.0.0"
  xmlns:rr="http://www.ibm.com/appserver/schemas/8.0/blueprint/resourcereference">
  <!-- Other Blueprint declarations ... -->

  <rr:resource-reference id="resourceRef"
          interface="javax.sql.DataSource"
          filter="(osgi.jndi.service.name=jdbc/Account)">
    <rr:res-auth>Container</rr:res-auth>
    <rr:res-sharing-scope>Shareable</rr:res-sharing-scope>
  </rr:resource-reference>
</blueprint>
```

   This declaration includes the resource reference ID (for example `resourceRef`), the service filter (for example `jdbc/Account`), the authentication type (for example `Container`), and the sharing setting (for example `Shareable`).

   The Blueprint resource references to authentication alias bindings for each bundle are stored in a file `ibm-eba-bnd.xml` in the `META-INF` directory of that bundle. If an OSGi application contains any of these files when it is deployed as an asset, these files provide the default authentication alias values that are used when binding the resource references. For example:

```
<eba-bnd xmlns="http://www.ibm.com/blueprintbinding.xsd">
  <resource-ref>
    <jndi-name>jdbc/Acount</jndi-name>
    <authentication-alias>Alias1</authentication-alias>
    <interface>javax.sql.DataSource</interface>
    <authentication>Container</authentication>
    <sharing-scope>Shareable</sharing-scope>
    <id>resourceRef</id>
  </resource-ref>
</eba-bnd>
```

   The jython syntax for this aspect of the command is as follows.

   Note: For composite bundles, the syntax is slightly different. See Step: Add authentication aliases to Blueprint resource references in composite bundles.

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -BlueprintResourceRefPostDeployStep [
    [
```

```
    bundle_symbolic_name
    bundle_version
    blueprint_resource_reference_id
    interface_name
    jndi_name
    authentication_type
    sharing_setting
    authentication_alias_name
    ]]
]')
```

Note: The value for *jndi_name* must match the jndi name that you declare in the **filter** attribute of the resource reference element in the Blueprint XML file.

For example, for an EBA file that contains a bundle `com.ibm.ws.eba.helloWorldService.properties.bundle.jar` at Version 1.0.0, which is to be bound to authentication alias `alias1`, the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -BlueprintResourceRefPostDeployStep[
    [com.ibm.ws.eba.helloWorldService.properties.bundle 1.0.0 resourceRef
    javax.sql.DataSource jdbc/Account Container Shareable alias1]]
]')
```

8. Bind web module message destination and resource environment references.

   Binding a resource reference maps a resource dependency of the web application to an actual resource available in the server runtime environment. At a minimum, this can be achieved by using a mapping that specifies the JNDI name under which the resource is known in the runtime environment. By default, the JNDI name is the resource ID that you specified in the `web.xml` file during development of the web application bundle (WAB). Use this option to bind resources of type message-destination-ref (message destination reference) or resource-env-ref (resource environment reference), as defined in the Java specification JSR-250: Common Annotations for the Java Platform.

   The jython syntax for this aspect of the command is as follows.

   Note: For composite bundles, the syntax is slightly different. See Step: Bind web module message destination and resource environment references in composite bundles.

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -WebModuleMsgDestRefs [
    [
    bundle_symbolic_name
    bundle_version
    resource_reference_id
    resource_type
    target_jndi_name
    ]]
]')
```

   For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -WebModuleMsgDestRefs [
    [com.ibm.ws.eba.helloWorldService.web 1.0.0
    jms/myQ javax.jms.Queue jms/workQ]
    [com.ibm.ws.eba.helloWorldService.web 1.0.0
    jms/myT javax.jms.Topic jms/notificationTopic]]
]')
```

9. Bind web module resource references.

Binding a resource reference maps a resource dependency of the web application to an actual resource available in the server runtime environment. At a minimum, this can be achieved by using a mapping that specifies the JNDI name under which the resource is known in the runtime environment. By default, the JNDI name is the resource ID that you specified in the `web.xml` file during development of the web application bundle (WAB). Use this option to bind resources of type resource-ref (resource reference), as defined in the Java specification JSR-250: Common Annotations for the Java Platform.

The jython syntax for this aspect of the command is as follows.

Note: For composite bundles, the syntax is slightly different. See Step: Bind web module resource references in composite bundles.

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -WebModuleResourceRefs [
    [
    bundle_symbolic_name
    bundle_version
    resource_reference_id
    resource_type
    target_jndi_name
    login_configuration
    login_properties
    extended_properties
    ]]
]')
```

For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -WebModuleResourceRefs [
    [com.ibm.ws.eba.helloWorldService.web 1.0.0
    jdbc/jtaDs javax.sql.DataSource
    jdbc/helloDs "" "" ""]
    [com.ibm.ws.eba.helloWorldService.web 1.0.0
    jdbc/nonJtaDs javax.sql.DataSource
    jdbc/helloDsNonJta "" "" "extprop1=extval1"]]
]')
```

Note: If you use multiple extended properties, the jython syntax is `"extprop1=extval1,extprop2=extval2"`.

- Configure the HTTP session manager.

  To configure the HTTP session manager, you use the AdminConfig commands to configure the deployed object represented by the *appDeploy* variable. Session management for OSGi applications is configured in the same way as for enterprise applications, except for a minor difference in syntax when getting the deployed object.

  1. Get the deployed object.

     Use the instructions given in Configuring applications for session management using scripting. Note that, for enterprise applications, you use the following two line script:

     ```
     deployments = AdminConfig.getid('/Deployment:myApp/')
     appDeploy = AdminConfig.showAttribute(deployments, 'deployedObject')
     ```

     For OSGi applications, the equivalent script is the following single line:

     ```
     appDeploy = AdminTask.getOSGiApplicationDeployedObject('-cuName cu_name')
     ```

     where *cu_name* is the name of the composition unit. For example:

```
appDeploy = AdminTask.getOSGiApplicationDeployedObject('
  -cuName com.ibm.ws.eba.helloWorldService_0001.eba')
```

2. Create or modify the session management options.

   Use the instructions given in Configuring applications for session management using scripting. The command usage for creating and modifying session management options is exactly the same for enterprise applications and OSGi applications.

What to do next

After using these commands, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

When you save the changes to the composition unit, the associated business-level application is updated to use the new configuration. If the business-level application is running, the bundle and configuration updates are applied immediately. If possible (that is, depending on the nature of the updates) the system applies the updates without restarting the application. If new packages are imported, or new services injected, then the application is restarted. These packages and services can come from newly-provisioned bundles, or from bundles that have already been provisioned.

Modifying the configuration of an OSGi composition unit that includes composite bundles using the editCompUnit command:

You can use the editCompUnit command to modify the configuration information for a composition unit that contains an enterprise bundle archive (EBA) asset. If the EBA asset includes composite bundles, the command syntax is slightly different.

Before you begin

For a full description of how you modify this configuration information, see “Modifying the configuration of an OSGi composition unit using wsadmin commands” on page 1065. When you work through that task, each step where the syntax is different for composite bundles is linked to an equivalent step in this task.

About this task

An OSGi composition unit includes an EBA asset and some or all of the following configuration information:

- Mappings from the composition unit to a target application server, web server or cluster.
- Configuration of the application's session manager, context roots or virtual hosts.
- Bindings to any associated web applications or blueprint resource references.
- Mappings from security roles to particular users or groups.

For several of the elements, the values you specify include bundle identifiers. If your EBA asset includes or references composite bundles, the command syntax is slightly different. The differences for composite bundles are described in the following steps.

Procedure

- Define context roots for web application bundles (WABs) in composite bundles.

  The jython syntax for this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
```

```
  -ContextRootStep [
    [bundle_symbolic_name_1 bundle_version_1 context_root_1]
    [bundle_symbolic_name_2 bundle_version_2 context_root_2]]
]')
```

For composite bundles, the bundle symbolic name has the following syntax:

```
CBA.symbolic.Name_CBA.bundle.version/WAB.symbolic.name
```

For example, for a composite bundle `com.ibm.ws.eba.helloWorldCBA` at version 1.0.0 that contains two WABs (`com.ibm.ws.eba.helloWorldService.web` at version 1.0.0, which is to be mapped to `/hello/web`, and `com.ibm.ws.eba.helloWorldService.withContextRoot` at version 0.9.0, which is to be mapped to `/hello/service`), this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -ContextRootStep [
    [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.web 1.0.0
    "/hello/web"]
    [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.withContextRoot 0.9.0
    "/hello/service"]]
]')
```

Note: For bundles other than composite bundles, the syntax is slightly different. See Step: Define WAB context roots.

- Map WABs in composite bundles to virtual hosts.

  The jython syntax for this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -VirtualHostMappingStep [
    [bundle_symbolic_name_1 bundle_version_1
    web_module_name_1 virtual_host_1]
    [bundle_symbolic_name_2 bundle_version_2
     web_module_name_2 virtual_host_2]]
]')
```

  For composite bundles, the bundle symbolic name has the following syntax:

```
CBA.symbolic.Name_CBA.bundle.version/WAB.symbolic.name
```

  For example, for a composite bundle `com.ibm.ws.eba.helloWorldCBA` at version 1.0.0 that contains two WABs (`com.ibm.ws.eba.helloWorldService.web` at version 1.0.0, which is to be mapped to `default_host`, and `com.ibm.ws.eba.helloWorldService.withContextRoot` at version 0.9.0, which is to be mapped to `test_host`), this aspect of the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -VirtualHostMappingStep [
    [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.web
    1.0.0 "HelloWorld service" default_host]
    [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.withContextRoot
    0.9.0
    "HelloWorld second service" test_host]]
]')
```

  Note: For bundles other than composite bundles, the syntax is slightly different. See Step: Map WABs to virtual hosts.

- Add authentication aliases to Blueprint resource references in composite bundles.

  The jython syntax for this aspect of the command is as follows.

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
```

```
  -BlueprintResourceRefPostDeployStep [
    [
    bundle_symbolic_name
    bundle_version
    blueprint_resource_reference_id
    interface_name
    jndi_name
    authentication_type
    sharing_setting
    authentication_alias_name
    ]]
]')
```

Notes:

- For composite bundles, the bundle symbolic name has the following syntax:

  ```
  CBA.symbolic.Name_CBA.bundle.version/inner_bundle.symbolic.name
  ```

- The value for *jndi_name* must match the jndi name that you declare in the **filter** attribute of the resource reference element in the Blueprint XML file.

For example, for a composite bundle `com.ibm.ws.eba.helloWorldCBA` at version 1.0.0 that contains a bundle `com.ibm.ws.eba.helloWorldService.properties.bundle.jar` at Version 1.0.0, which is to be bound to authentication alias `alias1`, the command is as follows:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -BlueprintResourceRefPostDeployStep[
    [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.properties.bundle
    1.0.0 resourceRef javax.sql.DataSource jdbc/Account Container Shareable alias1]]
]')
```

Note: For bundles other than composite bundles, the syntax is slightly different. See Step: Add authentication aliases to Blueprint resource references.

- Bind web module message destination and resource environment references in composite bundles.

  The jython syntax for this aspect of the command is as follows.

  ```
  AdminTask.editCompUnit('[
    -blaID WebSphere:blaname=bla_name
    -cuID WebSphere:cuname=cu_name
    -WebModuleMsgDestRefs [
      [
      bundle_symbolic_name
      bundle_version
      resource_reference_id
      resource_type
      target_jndi_name
      ]]
  ]')
  ```

  For composite bundles, the bundle symbolic name has the following syntax:

  ```
  CBA.symbolic.Name_CBA.bundle.version/WAB.symbolic.name
  ```

  For example:

  ```
  AdminTask.editCompUnit('[
    -blaID WebSphere:blaname=helloWorldService
    -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
    -WebModuleMsgDestRefs [
      [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.web
      1.0.0
      jms/myQ javax.jms.Queue
      jms/workQ]
      [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.web
  ```

```
    1.0.0
    jms/myT javax.jms.Topic
    jms/notificationTopic]]
]')
```

Note: For bundles other than composite bundles, the syntax is slightly different. See Step: Bind web module message destination and resource environment references.

- Bind web module resource references in composite bundles.

  The jython syntax for this aspect of the command is as follows.

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=bla_name
  -cuID WebSphere:cuname=cu_name
  -WebModuleResourceRefs [
    [
    bundle_symbolic_name
    bundle_version
    resource_reference_id
    resource_type
    target_jndi_name
    login_configuration
    login_properties
    extended_properties
    ]]
]')
```

  For composite bundles, the bundle symbolic name has the following syntax:

```
CBA.symbolic.Name_CBA.bundle.version/WAB.symbolic.name
```

  For example:

```
AdminTask.editCompUnit('[
  -blaID WebSphere:blaname=helloWorldService
  -cuID WebSphere:cuname=com.ibm.ws.eba.helloWorldService_0001.eba
  -WebModuleResourceRefs [
    [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.web
    1.0.0
    jdbc/jtaDs javax.sql.DataSource
    jdbc/helloDs "" "" ""]
    [com.ibm.ws.eba.helloWorldCBA_1.0.0/com.ibm.ws.eba.helloWorldService.web
    1.0.0
    jdbc/nonJtaDs javax.sql.DataSource
    jdbc/helloDsNonJta "" "" "extprop1=extval1"]]
]')
```

  Notes:

  – If you use multiple extended properties, the jython syntax is `"extprop1=extval1,extprop2=extval2"`.

  – For bundles other than composite bundles, the syntax is slightly different. See Step: Bind web module resource references.

Checking the bundle download status of an EBA asset

Updating an asset to use a new bundle version might require bundle downloads. You cannot update an asset until bundle downloads are complete from any previous update. Before you try and update bundle versions, you can check the bundle download status of the asset.

Before you begin

In addition to the approach given in this task, you can also check the bundle download status indirectly, by checking the status of the associated OSGi composition unit as described in “Checking the update status of an OSGi composition unit” on page 1077.

About this task

You use the administrative console to check the bundle download status of an EBA asset. This status is either “Bundles downloading...”, “Bundle downloads are complete”, or “No bundles downloads are required”.

Procedure

1. Start the administrative console.
2. Navigate to **Applications** > **Application Types** > **Assets** > ***asset_name***.
   The Asset settings panel is displayed.

Results

The current bundle download status for all bundles and composite bundles for this asset is displayed in the EBA Dependencies section. If the bundle downloads for any previous update have completed, the option to update bundle versions is available under the Additional Properties section.

What to do next

- If bundle downloads for the asset are complete, or no bundle downloads are required, you can update the asset as described in “Updating bundle versions for an EBA asset” on page 1048.
- If a new version of the EBA asset is available, and all bundle downloads for the asset are complete, you can update the OSGi composition unit so that the business-level application uses the newer configuration. See “Updating an OSGi composition unit” on page 1052.

Checking the update status of an OSGi composition unit

When you change the versions of bundles or composite bundles that an enterprise OSGi application uses, or add or remove a composite bundle extension, the bundles used by the deployed application can get out of synchronization with those that are available. To check that your business-level application is running the most recent version of the EBA asset and any composite bundle extensions, you check the update status of the associated OSGi composition unit.

About this task

When an EBA asset is updated, or a composite bundle extension is added or replaced, the update does not automatically affect the running business-level application. To update the running application, you update the composition unit of the business-level application that contains the asset.

The deployment status shows whether an updated version is available of the EBA asset or composite bundle extension that is contained in the composition unit. There are four distinct deployment statuses for an OSGi composition unit:

Using latest OSGi application deployment.
The composition unit is running the latest configuration of the backing asset and any composite bundle extensions.

New OSGi application deployment not yet available because it requires bundles that are still downloading.
The bundle downloads that were started after you last updated an asset, or added a composite bundle extension, have not yet finished.

New OSGi application deployment available.
The backing asset is available at a newer configuration than the configuration that is currently running in this composition unit, or a composite bundle extension has been added or replaced.

New OSGi application deployment cannot be applied because bundle downloads have failed.
The bundle downloads that were started after you last updated an asset, or added a composite

bundle extension, have not succeeded. Resolve the problem, then download the bundles again. See “Interacting with the OSGi bundle cache” on page 1095.

You can check whether the OSGi composition unit is up to date by using the administrative console, or by using the viewCompUnit command.

Procedure

- Check the update status of the OSGi composition unit using the administrative console.
  1. Start the administrative console.
  2. Navigate to **Applications** > **Application Types** > **Business-level applications** > ***application_name*** > ***composition_unit_name***.

  The Composition unit settings panel is displayed. The deployment status is displayed under **[General Properties] OSGi application deployment status**, and shows one of the following values:
  - Using latest OSGi application deployment.
  - New OSGi application deployment not yet available because it requires bundles that are still downloading.
  - New OSGi application deployment available.
  - New OSGi application deployment cannot be applied because bundle downloads have failed.
- Check the update status of the OSGi composition unit using the viewCompUnit command.
  1. Open a wsadmin command prompt.
  2. Run the following jython command:

     ```
     AdminTask.viewCompUnit('[-blaID WebSphere:blaname=bla_name
                             -cuID WebSphere:cuname=cu_name ]')
     ```

     For more information about using the viewCompUnit command, see the step “Display composition units and configuration settings” in topic Managing composition units using wsadmin scripting.

  The set of information about the OSGi composition unit is displayed, including the composition unit status. This status is one of the following values:
  - Using latest OSGi application deployment.
  - New OSGi application deployment not yet available because it requires bundles that are still downloading.
  - New OSGi application deployment available.
  - New OSGi application deployment cannot be applied because bundle downloads have failed.

What to do next

- If there is no update in progress, or bundle downloads for the asset have not succeeded, you can update the asset as described in “Updating bundle versions for an EBA asset” on page 1048.
- If a new OSGi application deployment is available, you can update the OSGi composition unit so that the business-level application uses the newer configuration. See “Updating an OSGi composition unit” on page 1052.

Administering bundle repositories

OSGi applications can share a large number of common utility bundles. To simplify maintenance, and reduce the application footprint, the application does not need to include its own copy of each utility bundle. Instead, bundles can be hosted in a bundle repository, from where they are retrieved during deployment. You can administer bundles and composite bundles held in the internal bundle repository of the product, and also add links to external bundle repositories.

About this task

WebSphere Application Server includes an internal bundle repository, in which you can store the bundles and composite bundles for your OSGi applications. The external bundle repositories are bundle repositories that are available outside of WebSphere Application Server. If your OSGi applications reference bundles that are stored in an external bundle repository, you must configure a link (name and URL) to the repository so that the provisioner can retrieve the bundles when required. When an OSGi application is imported as an asset, the provisioner attempts to satisfy all its dependencies by using the contents of the asset, the contents of the internal bundle repository, and the contents of any available external bundle repositories.

You can link to any external bundle repository that complies with the OSGi Alliance RFC-0112 Bundle Repository Draft Specification. Such a repository consists of a website with a bundle repository XML file that describes an OSGi-compliant repository. Depending on how the external bundle repository is implemented, you might not be able to use it to provision services, or to store composite bundles or bundles referenced by composite bundles.

If your bundle includes Blueprint XML files that specify service or reference elements, and the bundle is included in a EBA asset or installed in the internal bundle repository, then these elements are respected during provisioning and appropriate services are provisioned when needed. For more information, see Provisioning for OSGi applications.

Procedure

- Move bundles from an OSGi application to a bundle repository.

  You can change the configuration of an enterprise OSGi application so that bundles and composite bundles that were directly contained in the enterprise bundle archive (EBA) file are instead pulled in by reference and provisioned from a repository. Before you deploy the application, you can move bundles or composite bundles from the EBA file to a bundle repository. You might want to do this to decrease the size of the EBA file, or so that bundles can be shared between multiple applications rather than each application deploying its own copy of a common library.
- List, delete, add, or show details for bundles and composite bundles in the internal bundle repository.

  Do this using the administrative console or wsadmin commands:
  - “Administering bundles in the internal bundle repository” on page 1080
  - “Administering bundles in the internal bundle repository using wsadmin commands” on page 1081
- List, delete, add, modify, or show details for links to external bundle repositories.

  Do this using the administrative console or wsadmin commands:
  - “Administering links to external bundle repositories” on page 1088
  - “Administering links to external bundle repositories using wsadmin commands” on page 1089
- Interact with the OSGi bundle cache.

Moving bundles from an OSGi application to a bundle repository

You can change the configuration of an enterprise OSGi application so that bundles and composite bundles that were directly contained in the enterprise bundle archive (EBA) file are instead pulled in by reference and provisioned from a repository.

Before you begin

After you import your application as an asset, you can no longer change the direct contents of the asset.To update bundles and composite bundles that are specified in an asset, you add the updated versions to a repository then apply the updates to the asset. See Updating bundle versions in a deployed OSGi application.

About this task

When you first create an OSGi application, each bundle and composite bundle in the application is either directly contained in the EBA file or pulled in by reference. Before you deploy the application, you can move bundles or composite bundles from the EBA file to a bundle repository. You might want to do this to decrease the size of the EBA file, or so that bundles can be shared between multiple applications rather than each application deploying its own copy of a common library.

Composite bundles, and any bundles that are used by composite bundles, can be moved to the internal bundle repository or to an external repository that can process composite bundles. You can move the whole composite bundle to the repository, or move one or more of the bundles that the composite bundle uses to the repository. For more information, see Composite bundles.

Procedure

1. Use your preferred development tool to remove the bundle or composite bundle from the EBA file.
   See Developing and deploying an OSGi application.
2. Ensure that the EBA file has an application manifest and that the bundle or composite bundle is listed in the Application-Content header of the application manifest.
   See Enterprise bundle archives.
3. Install the bundle that you removed into a bundle repository.
   Composite bundles, and bundles referenced by composite bundles, are installed in the internal bundle repository or in an external repository that can process composite bundles. See “Administering bundle repositories” on page 1078.

What to do next

You can deploy your OSGi application. See Deploying an OSGi application as a business-level application.

Administering bundles in the internal bundle repository

Use the administrative console to list, delete, add, or show further details for bundles and composite bundles that are held in the bundle repository that is included in the product.

Before you begin

You can administer bundles and composite bundles in the internal bundle repository by using the administrative console as described in this topic, or by using wsadmin commands as described in “Administering bundles in the internal bundle repository using wsadmin commands” on page 1081.

About this task

WebSphere Application Server includes an internal bundle repository, in which you can store the bundles and composite bundles for your OSGi applications.

If your OSGi applications are configured to expect to find certain bundles in the internal bundle repository, you must add those bundles to the repository. Composite bundles can either be included directly in your applications, or provisioned from the internal bundle repository or from an external repository that can process composite bundles. If your bundle includes Blueprint XML files that specify service or reference elements, and the bundle is included in a EBA asset or installed in the internal bundle repository, then these elements are respected during provisioning and appropriate services are provisioned when needed. For more information, see Provisioning for OSGi applications.

You can install bundles singly, or you can install a set of bundles packaged as a compressed archive file with a `.zip` file extension. In both cases, the bundles are available individually in the repository. If you install a composite bundle in a bundle repository, and the composite bundle includes bundles by reference,

you must ensure that the referenced bundles are also available in the same repository. If you use the internal bundle repository, and the composite bundle directly contains bundles, the contained bundles are not listed separately and are only available as part of the composite bundle. For more information, see Composite bundles.

You can list, delete, add, or show further details for bundles and composite bundles that are held in the internal bundle repository.

Procedure

1. Start the administrative console.
2. Navigate to **Environment** > **OSGi bundle repositories** > **Internal bundle repository**.

   The list of bundles and composite bundles stored in the internal bundle repository is displayed in the Internal bundle repository [Collection] form.
3. Delete, add, or show details of bundles that are held in the internal bundle repository.
   - To delete one or more bundles from the repository, select the required bundles then click **Delete**.

     You cannot delete a bundle if it is referenced by a composite bundle. You must delete the composite bundle first, then delete the bundle.
   - To add a new bundle or composite bundle to the repository:
     a. Click **New**. The Upload bundle [Settings] form is displayed.
     b. Enter the path to the bundle, composite bundle or grouped-up set of bundles that you want to add. Each individual bundle must be packaged as a `.jar` file, and must contain a suitably-configured bundle manifest file. Each composite bundle must be packaged as a compressed archive file with a `.cba` file extension, and must contain a suitably-configured composite bundle manifest file. Each grouped-up set of bundles must be packaged as a compressed archive file with a `.zip` file extension.
     c. Click **OK**. The file is uploaded, and displayed in the list of bundles and composite bundles in the Internal bundle repository [Collection] form.
   - To show the details for an existing bundle or composite bundle in the repository, click the name of the bundle or composite bundle. The details are displayed in the Internal bundle repository [Settings] form. You cannot modify these details.
4. If you added or removed a bundle or composite bundle, save your changes to the master configuration.

What to do next

- If you updated a bundle or composite bundle, and you want to update an asset that uses it, see “Updating bundle versions for an EBA asset” on page 1048.
- If you added or updated a bundle or composite bundle, and you want to install an enterprise OSGi application that uses it, see Deploying an OSGi application as a business-level application.
- If you added or updated a composite bundle, and you want to use it to extend a composition unit, see “Adding or removing extensions for an OSGi composition unit” on page 1055.

Administering bundles in the internal bundle repository using wsadmin commands

Use wsadmin commands to list, add, remove, or show further details for bundles and composite bundles that are held in the bundle repository that is included in the product.

Before you begin

You can administer bundles and composite bundles in the internal bundle repository by using wsadmin commands as described in this topic, or by using the administrative console as described in “Administering bundles in the internal bundle repository” on page 1080.

About this task

WebSphere Application Server includes an internal bundle repository, in which you can store the bundles and composite bundles for your OSGi applications.

If your OSGi applications are configured to expect to find certain bundles in the internal bundle repository, you must add those bundles to the repository. Composite bundles can either be included directly in your applications, or provisioned from the internal bundle repository or from an external repository that can process composite bundles. If your bundle includes Blueprint XML files that specify service or reference elements, and the bundle is included in a EBA asset or installed in the internal bundle repository, then these elements are respected during provisioning and appropriate services are provisioned when needed. For more information, see Provisioning for OSGi applications.

You can install bundles singly, or you can install a set of bundles packaged as a compressed archive file with a `.zip` file extension. In both cases, the bundles are available individually in the repository. If you install a composite bundle in a bundle repository, and the composite bundle includes bundles by reference, you must ensure that the referenced bundles are also available in the same repository. If you use the internal bundle repository, and the composite bundle directly contains bundles, the contained bundles are not listed separately and are only available as part of the composite bundle. For more information, see Composite bundles.

You can list, add, remove, or show further details for bundles and composite bundles that are held in the internal bundle repository.

Procedure

- List all bundles and composite bundles that are held in the internal bundle repository.

  Use the listLocalRepositoryBundles command. For example:

  ```
  AdminTask.listLocalRepositoryBundles()
  ```

  For more information, see “listLocalRepositoryBundles command” on page 1083.

  Notes:

  - The list includes any bundles that you have added since you last saved your changes, and excludes any bundles that you have removed since you last saved your changes.
  - The output from this command is formatted so that you can copy the list of bundles, then paste them into the removeLocalRepositoryBundles command.

- Add a bundle, composite bundle or grouped-up set of bundles to the internal bundle repository.

  Use the addLocalRepositoryBundle command. For example:

  ```
  AdminTask.addLocalRepositoryBundle('-file path')
  ```

  Each individual bundle must be packaged as a `.jar` file, and must contain a suitably-configured bundle manifest file. Each composite bundle must be packaged as a compressed archive file with a `.cba` file extension, and must contain a suitably-configured composite bundle manifest file. Each grouped-up set of bundles must be packaged as a compressed archive file with a `.zip` file extension.

  For more information, see “addLocalRepositoryBundle command” on page 1084.

- Remove one or more bundles or composite bundles from the internal bundle repository.

  Use the removeLocalRepositoryBundle or removeLocalRepositoryBundles command. For example:

  ```
  AdminTask.removeLocalRepositoryBundle('-symbolicName bundle_symbolic_name
                                        -version bundle_version')
  ```

```
AdminTask.removeLocalRepositoryBundles([
    'bundle1_symbolic_name;bundle1_version
     bundle2_symbolic_name;bundle2_version
     bundle3_symbolic_name;bundle3_version
    '
])
```

Notes:

- The output from the listLocalRepositoryBundles command is formatted so that you can copy the list of bundles, then paste them into the removeLocalRepositoryBundles command.
- You cannot remove composite bundles at the same time as you remove any bundles that they reference. You must first remove the composite bundles, then run the removeLocalRepositoryBundles command a second time to remove the referenced bundles.

For more information, see “removeLocalRepositoryBundle command” on page 1085 and “removeLocalRepositoryBundles command” on page 1086.

- Show further details (for example, imported packages, exported packages, and required bundles) for a bundle or composite bundle in the internal bundle repository.

  Use the showLocalRepositoryBundle command. For example:

  ```
  AdminTask.showLocalRepositoryBundle('-symbolicName bundle_symbolic_name
                                      -version bundle_version')
  ```

  For more information, see “showLocalRepositoryBundle command” on page 1087.

What to do next

- If you added or removed a bundle or composite bundle, save your changes to the master configuration.
- If you updated a bundle or composite bundle, and you want to update an asset that uses it, see “Updating bundle versions for an EBA asset using the editAsset command” on page 1049.
- If you added or updated a bundle or composite bundle, and you want to install an enterprise OSGi application that uses it, see Deploying an OSGi application as a business-level application.
- If you added or updated a composite bundle, and you want to use it to extend a composition unit, see “Adding or removing extensions for an OSGi composition unit using wsadmin commands” on page 1056..

listLocalRepositoryBundles command:

Use the listLocalRepositoryBundles command to list all bundles held in the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all bundles that are held in the internal bundle repository. The list includes any bundles that you have added since you last saved your changes, and excludes any bundles that you have removed since you last saved your changes.

Target object

None.

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listLocalRepositoryBundles()
```

addLocalRepositoryBundle command:

Use the addLocalRepositoryBundle command to add a bundle, composite bundle or grouped-up set of bundles to the internal bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a bundle, a composite bundle or a grouped-up set of bundles to the internal bundle repository.

You can install bundles singly, or you can install a set of bundles packaged as a compressed archive file with a `.zip` file extension. In both cases, the bundles are available individually in the repository. If you install a composite bundle in a bundle repository, and the composite bundle includes bundles by reference, you must ensure that the referenced bundles are also available in the same repository. If you use the

internal bundle repository, and the composite bundle directly contains bundles, the contained bundles are not listed separately and are only available as part of the composite bundle. For more information, see Composite bundles.

Target object

None

Required parameters

-file *path*
The path and file name of a compressed archive file that has a `.jar`, `.cba` or `.zip` file extension and is available on the server file system.

Each individual bundle must be packaged as a `.jar` file, and must contain a suitably-configured bundle manifest file. Each composite bundle must be packaged as a compressed archive file with a `.cba` file extension, and must contain a suitably-configured composite bundle manifest file. Each grouped-up set of bundles must be packaged as a compressed archive file with a `.zip` file extension.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.addLocalRepositoryBundle('-file path')
```

removeLocalRepositoryBundle command:

Use the removeLocalRepositoryBundle command to remove a bundle or composite bundle from the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes a bundle or composite bundle from the internal bundle repository.

You cannot delete a bundle if it is contained in, or referenced by, a composite bundle. You must delete the composite bundle first, then delete the bundle.

Target object

The specified bundle.

Required parameters

-symbolicName *bundle_symbolic_name*
One of the following:
- The non-localizable name for this bundle.
- The bundle symbolic name of this composite bundle.

The bundle symbolic name, together with the bundle version, identifies a unique bundle.

-version *bundle_version*
One of the following:
- The version of this bundle.
- The version of this composite bundle.

The bundle version is in the form *n.n.n*, for example `1.1.0`. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

Note: You can use the listLocalRepositoryBundles command to list the symbolic names and versions of the bundles currently held in the repository.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeLocalRepositoryBundle('-symbolicName bundle_symbolic_name
                                       -version bundle_version')
```

removeLocalRepositoryBundles command:

Use the removeLocalRepositoryBundles command to remove bundles and composite bundles from the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:
- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes bundles and composite bundles from the internal bundle repository.

You cannot remove composite bundles at the same time as you remove any bundles that they reference. You must first remove the composite bundles, then run the removeLocalRepositoryBundles command a second time to remove the referenced bundles.

You can use the listLocalRepositoryBundles command to list the symbolic names and versions of the bundles currently held in the repository. The output from the listLocalRepositoryBundles command is formatted so that you can copy the list of bundles, then paste them into the removeLocalRepositoryBundles command.

Target object

The specified bundles.

Required parameters

A list of the bundles to be removed.

The list entry for each bundle or composite bundle contains the bundle symbolic name and the bundle version. The bundle version is in the form *n.n.n*, for example `1.1.0`. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeLocalRepositoryBundles([
    'bundle1_symbolic_name;bundle1_version
     bundle2_symbolic_name;bundle2_version
     bundle3_symbolic_name;bundle3_version
    '
])
```

showLocalRepositoryBundle command:

Use the showLocalRepositoryBundle command to show further details (in particular all manifest header entries) for a bundle in the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

This command shows the values of the following properties for the specified bundle stored in the internal bundle repository:

- Bundle symbolic name
- Bundle version
- Bundle name
- Bundle description
- Imported packages
- Exported packages
- Required bundles

For more information about these properties, see Internal bundle repository [Settings].

Target object

The specified bundle.

Required parameters

-symbolicName *bundle_symbolic_name*
The non-localizable name for this bundle. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

-version *bundle_version*
The version of this bundle. The bundle version is in the form *n.n.n*, for example `1.1.0`. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

Note: You can use the listLocalRepositoryBundles command to list the symbolic names and versions of the bundles currently held in the repository.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.showLocalRepositoryBundle('-symbolicName bundle_symbolic_name
                                    -version bundle_version')
```

Administering links to external bundle repositories

Use the administrative console to list, delete, add, modify, or show details for links to external bundle repositories.

Before you begin

You can administer links to external bundle repositories by using the administrative console as described in this topic, or by using wsadmin commands as described in “Administering links to external bundle repositories using wsadmin commands.”

About this task

The external bundle repositories are bundle repositories that are available outside of WebSphere Application Server. If your OSGi applications reference bundles that are stored in an external bundle repository, you must configure a link (name and URL) to the repository so that the provisioner can retrieve the bundles when required. When an OSGi application is imported as an asset, the provisioner attempts to satisfy all its dependencies by using the contents of the asset, the contents of the internal bundle repository, and the contents of any available external bundle repositories.

Depending on how the external bundle repository is implemented, you might not be able to use it to provision services, or to store composite bundles or bundles referenced by composite bundles. If your bundle includes Blueprint XML files that specify service or reference elements, and the bundle is included in a EBA asset or installed in the internal bundle repository, then these elements are respected during provisioning and appropriate services are provisioned when needed. For more information, see Provisioning for OSGi applications.

Procedure

1. Start the administrative console.
2. Navigate to **Environment** > **OSGi bundle repositories** > **External bundle repositories**.

   The list of external bundle repository links is displayed in the External bundle repositories [Collection] form.
3. Delete, add, show, or modify links to external bundle repositories.
   - To delete one or more repository links from the list, select the required links then click **Delete**.
   - To add a new link to an external bundle repository, click **New** then type the link properties (name, optional description, and URL of the bundle repository XML file) into the External bundle repositories [Settings] form.
   - To show or modify an existing link to an external bundle repository, click the name of the link. The link properties are displayed. If required, modify the link properties (optional description, and URL of the bundle repository XML file) in the External bundle repositories [Settings] form. You cannot modify the **bundle repository name** property.
4. If you add, modify, or remove a link to an external bundle repository, save your changes to the master configuration.

   If you add or modify a link to an external bundle repository, you must save the changes before you can install an enterprise bundle archive (EBA) that depends on a bundle in that repository.

Administering links to external bundle repositories using wsadmin commands

Use wsadmin commands to list, remove, add, modify, or show details for links to external bundle repositories.

Before you begin

You can administer links to external bundle repositories by using wsadmin commands as described in this topic, or by using the administrative console as described in “Administering links to external bundle repositories” on page 1088.

About this task

The external bundle repositories are bundle repositories that are available outside of WebSphere Application Server. If your OSGi applications reference bundles that are stored in an external bundle repository, you must configure a link (name and URL) to the repository so that the provisioner can retrieve the bundles when required. When an OSGi application is imported as an asset, the provisioner attempts to satisfy all its dependencies by using the contents of the asset, the contents of the internal bundle repository, and the contents of any available external bundle repositories.

Depending on how the external bundle repository is implemented, you might not be able to use it to provision services, or to store composite bundles or bundles referenced by composite bundles. If your bundle includes Blueprint XML files that specify service or reference elements, and the bundle is included in a EBA asset or installed in the internal bundle repository, then these elements are respected during provisioning and appropriate services are provisioned when needed. For more information, see Provisioning for OSGi applications.

Procedure

- List all links to external bundle repositories.

  Use the listExternalBundleRepositories command. For example:

  ```
  AdminTask.listExternalBundleRepositories()
  ```

  For more information, see “listExternalBundleRepositories command” on page 1091.

  Note: The list includes any repository links that you have added since you last saved your changes, and excludes any repository links that you have removed since you last saved your changes.
- Show the configured parameters of an external bundle repository.

  Use the showExternalBundleRepository command. For example:

  ```
  AdminTask.showExternalBundleRepository('-name bundle_repository_name')
  ```

  For more information, see “showExternalBundleRepository command” on page 1094.
- Remove a link to an external bundle repository.

  Use the removeExternalBundleRepository command. For example:

  ```
  AdminTask.removeExternalBundleRepository('-name bundle_repository_name')
  ```

  For more information, see “removeExternalBundleRepository command” on page 1091.
- Add a link to an external bundle repository.

  Use the addExternalBundleRepository command. For example:

  ```
  AdminTask.addExternalBundleRepository('-name bundle_repository_name
                                 -url bundle_repository_URL
                                [-description bundle_repository_description]')
  ```

  Square brackets (“[ ]”) indicate that a parameter is optional.

  For more information, see “addExternalBundleRepository command” on page 1092.
- Modify a link to an external bundle repository.

  Use the modifyExternalBundleRepository command. For example:

  ```
  AdminTask.modifyExternalBundleRepository('-name bundle_repository_name
                                [-url bundle_repository_URL]
                                [-description bundle_repository_description]')
  ```

  Square brackets (“[ ]”) indicate that a parameter is optional.

  For more information, see “modifyExternalBundleRepository command” on page 1093.

What to do next

If you add, modify, or remove a link to an external bundle repository, save your changes to the master configuration.

If you add or modify a link to an external bundle repository, you must save the changes before you can install an enterprise bundle archive (EBA) that depends on a bundle in that repository.

listExternalBundleRepositories command:

Use the listExternalBundleRepositories command to list all links to external bundle repositories.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all links to external bundle repositories. The list includes any repository links that you have added since you last saved your changes, and excludes any repository links that you have removed since you last saved your changes.

Target object

None.

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listExternalBundleRepositories()
```

removeExternalBundleRepository command:

Use the removeExternalBundleRepository command to remove a link to an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes a link to an external bundle repository.

Target object

The specified bundle repository link.

Required parameters

-name *bundle_repository_name*
The name by which the bundle repository link is known. You can use the listExternalBundleRepositories command to list the names of existing bundle repository links.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeExternalBundleRepository('-name bundle_repository_name')
```

addExternalBundleRepository command:

Use the addExternalBundleRepository command to add a link to an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a link to an external bundle repository.

Target object

None

Required parameters

-name *bundle_repository_name*
The name by which you want the external bundle repository link to be known.

-url *bundle_repository_URL*
The URL of the bundle repository XML file. For example, `http://external_location/repository.xml`.

Conditional parameters

None.

Optional parameters

-description *bundle_repository_description*
An optional description of the bundle repository.

Example

```
AdminTask.addExternalBundleRepository('-name bundle_repository_name
                                -url http://external_location/repository.xml
                               [-description bundle_repository_description]')
```

Square brackets (“[ ]”) indicate that a parameter is optional.

modifyExternalBundleRepository command:

Use the modifyExternalBundleRepository command to modify a link to an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command modifies a link to an external bundle repository.

Target object

The specified external bundle repository link.

Required parameters

-name *bundle_repository_name*
The name of the external bundle repository link. You cannot change this value. You can use the listExternalBundleRepositories command to list the names of existing bundle repository links.

Conditional parameters

None.

Optional parameters

-url *bundle_repository_URL*
The modified URL for the external bundle repository XML file.

-description *bundle_repository_description*
The modified description of the external bundle repository link.

Example

```
AdminTask.modifyExternalBundleRepository('-name bundle_repository_name
                              [-url http://external_location/repository.xml]
                              [-description bundle_repository_description]')
```

Square brackets ("[ ]") indicate that a parameter is optional.

showExternalBundleRepository command:

Use the showExternalBundleRepository command to show the configured parameters of an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command shows the name, description when available, and URL of the bundle repository XML file, of an external bundle repository.

Target object

None.

Required parameters

-name *repository_name*
The name of a link to an external bundle repository.

Note: You can use the listExternalBundleRepositories command to list all links to external bundle repositories.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.showExternalBundleRepository(-name ExternalRepository1)
```

Interacting with the OSGi bundle cache

The bundle cache is a cell-wide store (or server-wide for single-server systems) of bundles that are referenced by OSGi applications, and that have been downloaded from both internal and external repositories. You can interact with the bundle cache using either the administrative console, or the methods of the OSGi BundleCacheManager MBean.

About this task

You can get an up-to-date list of the bundles in the bundle cache, check if all bundles have been successfully downloaded, and request that one or more bundles be downloaded again. For a given bundle you can view the bundle size, the download status, and a list of the assets and composition units that use the bundle. You can also view and refresh the repository URL for the repository that hosts the bundle.

The main differences between interacting with the OSGi bundle cache using the administrative console and using the MBean interface are as follows:

- You can use the Bundle cache [Collection] panel to change the sort order in the table, and to filter by (for example) bundle name, or by bundle state.
- You can use the MBean interface to remove a bundle from the cache.

The methods for the OSGi BundleCacheManager MBean interface are documented in the Additional Application Programming Interfaces (APIs) section of the generated API documentation.

You use the MBean methods with the AdminControl object of the wsadmin scripting client. For example:

```
objNameString = AdminControl.completeObjectName('WebSphere:type=BundleCacheManager,*')
print AdminControl.invoke(objNameString, 'areAllDownloadsComplete')
```

For more information about using MBean methods with the AdminControl object, see the “invoke” and “invoke_jmx” sections of Commands for the AdminControl object using wsadmin scripting.

Some common tasks for which you might interact with the bundle cache are as follows:

Procedure

- Interrogate the state of bundles.

  You can use the MBean interface to complete the following tasks:

  – Check the state of a specific bundle.
  – List the completed downloads.
  – List the unsuccessful downloads.

  Similarly, the state of all bundles is displayed on the Bundle cache [Collection] panel. This is one of the following states:

  Unknown
  The bundle is not in the bundle cache and there has been no request to download the bundle.

  Download requested
  A request has been issued to download the bundle, but the download has not yet begun.

  Downloading
  The bundle is downloading.

  Downloaded
  The bundle download is complete.

  Failed The bundle download has failed.

  Unsaved
  The bundle is unsaved if you have imported an asset and not yet saved your changes to the master configuration.

  The bundle is not downloaded until you save your changes.

- Check that all bundle downloads are complete before adding an enterprise bundle archive (EBA) asset to a business-level application.

  If all bundle downloads are complete, the state of every bundle is displayed as “Downloaded” on the Bundle cache [Collection] panel.

  You can use scripting to import an EBA file as an asset, then add the EBA asset to a business-level application. However, you cannot add the asset to the application until all the bundles are downloaded. You might therefore choose to code your script to call the areAllDownloadsComplete () method, then wait until the method confirms that all bundles are downloaded before adding the EBA asset to the business-level application.

- Resolve an unsuccessful bundle download.

  If a bundle does not download, complete the following steps:

  1. Fix the cause (for example an incorrect repository address, or a network failure).

     The Bundle cache [Settings] form for the bundle shows the repository address, and includes a button to refresh this address. This is useful if a bundle has been moved, for example from the internal bundle repository to an external repository. If the bundle state is “Failed”, the “Bundle download exceptions” pane is displayed. This pane contains trace information to help you understand why the bundle download has failed.

  2. Download the bundle again.

     Use either of the following approaches:

     – Use the resetBundleDownload () method to make the bundle available for download again, then use the downloadBundles () method to retry the download.
     – On the Bundle cache [Collection] panel, select one or more bundles then click **Download Bundle Again**.

- Remove a bundle from the cache.

  You can only do this using the MBean interface. Before you do this, make sure that no application is running that uses this bundle.

Use the removeBundleFromCache () method to remove a bundle from the bundle cache.

Exporting and importing a deployment manifest file

You can export the deployment manifest file from an application, then import the manifest file into another instance of the same application located somewhere else. This process is useful when an application moves from one environment to another, for example from a test environment to a production environment.

About this task

A deployment manifest file, `META-INF/DEPLOYMENT.MF`, is created automatically when you import an EBA asset. The deployment manifest file lists, at specific versions, all the bundles and composite bundles that make up the application, including bundles that are determined following dependency analysis. The manifest file is used to ensure that each time an application server starts, the bundles that make up the application are the same.

You can export the current deployment manifest from an EBA asset, then import the deployment manifest into another asset that contains the same application. The target asset then uses the imported manifest instead of the generated manifest. This is useful during application development, when an application is fully tested and moves to a production environment. By importing the deployment manifest from the test environment, you ensure that the bundles and their versions that make up the application in the production environment are exactly the same as the bundles that make up the application in the test environment.

Note: Do not edit an exported manifest file. Only use the export and import options in situations where you can treat the exported file as a “black box”.

Procedure

- Export the deployment manifest file from an EBA asset.

  You might want to do this to save the information, or to import it into another identical application.
- Import the deployment manifest file to an EBA asset.

  When you import the file, the bundles are resolved. If the bundles cannot be resolved, the import does not complete and an exception message is generated.

Exporting a deployment manifest

You can export the current deployment manifest file from an enterprise bundle archive (EBA) asset. You might want to do this to save the information, or to import it into another identical application.

Before you begin

You can export a deployment manifest by using the administrative console as described in this topic, or by using wsadmin commands, as described in “Exporting a deployment manifest using the exportDeploymentManifest command” on page 1098.

About this task

A deployment manifest file, `META-INF/DEPLOYMENT.MF`, is created automatically when you import an EBA asset. The deployment manifest file lists, at specific versions, all the bundles and composite bundles that make up the application, including bundles that are determined following dependency analysis. The manifest file is used to ensure that each time an application server starts, the bundles that make up the application are the same.

You can export the current deployment manifest from an EBA asset, then import the deployment manifest into another asset that contains the same application. The target asset then uses the imported manifest instead of the generated manifest. This is useful during application development, when an application is

fully tested and moves to a production environment. By importing the deployment manifest from the test environment, you ensure that the bundles and their versions that make up the application in the production environment are exactly the same as the bundles that make up the application in the test environment. Do not edit an exported manifest file. Only use the export and import options in situations where you can treat the exported file as a “black box”.

Procedure

1. Start the administrative console.
2. Navigate to **Applications** > **Application Types** > **Assets**.
3. Click the name of the asset from which you want to export the deployment manifest. The Asset settings panel is displayed.
4. Click **[Additional Properties] Export the deployment manifest from this application**.
5. Use the web browser dialog that is displayed to save the file to your required location. If the dialog does not offer the options that you need, check you web browser options, for example the options to download or save files.

Results

The current deployment manifest for the application is exported.

What to do next

The deployment manifest is available to import into an EBA asset. See “Importing a deployment manifest” on page 1100.

Exporting a deployment manifest using the exportDeploymentManifest command

You can use this command to export the current deployment manifest file from an enterprise bundle archive (EBA) asset. You might want to do this to save the information, or to import it into another identical application.

Before you begin

You can export a deployment manifest by using wsadmin commands, as described in this topic, or by using the administrative console, as described in “Exporting a deployment manifest” on page 1097.

About this task

A deployment manifest file, `META-INF/DEPLOYMENT.MF`, is created automatically when you import an EBA asset. The deployment manifest file lists, at specific versions, all the bundles and composite bundles that make up the application, including bundles that are determined following dependency analysis. The manifest file is used to ensure that each time an application server starts, the bundles that make up the application are the same.

You can export the current deployment manifest from an EBA asset, then import the deployment manifest into another asset that contains the same application. The target asset then uses the imported manifest instead of the generated manifest. This is useful during application development, when an application is fully tested and moves to a production environment. By importing the deployment manifest from the test environment, you ensure that the bundles and their versions that make up the application in the production environment are exactly the same as the bundles that make up the application in the test environment. Do not edit an exported manifest file. Only use the export and import options in situations where you can treat the exported file as a “black box”.

Procedure

1. Start the wsadmin scripting client if it is not already running.
2. Use the exportDeploymentManifest command. For example:

```
AdminTask.exportDeploymentManifest('[-asset
com.ibm.ws.eba.example.blabber.app.eba -path /test/temp/]')
```

Results

The current deployment manifest for the application is exported.

What to do next

The deployment manifest is available to import into an EBA asset. See “Importing a deployment manifest using the importDeploymentManifest command” on page 1101.

exportDeploymentManifest command:

Use the exportDeploymentManifest command to export the current deployment manifest, the `DEPLOYMENT.MF` file, from an enterprise bundle archive (EBA) asset. You might want to do this to save the information, or to import it into another identical application.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command exports the current deployment manifest from an EBA asset.

Target object

None.

Required parameters

-asset
The name of the EBA asset to export the deployment manifest from. This must be an installed EBA asset.

-path
The file path for the location of the exported deployment manifest. If the location does not exist, the export process creates it.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.exportDeploymentManifest('[-asset
com.ibm.ws.eba.example.blabber.app.eba -path /test/temp/]')
```

Importing a deployment manifest

You can import a suitable deployment manifest to an enterprise bundle archive (EBA) asset. When you import the file, the bundles are resolved. If the bundles cannot be resolved, the import does not complete and an exception message is generated.

Before you begin

The file to import must be a valid deployment manifest file, using the naming format *file_name*.MF, for example DEPLOYMENT_TEST.MF. When you import the deployment manifest into the EBA asset, the file is renamed to DEPLOYMENT.MF.

For the import to succeed, the following conditions must be met:

- The deployment manifest to import must correspond with the application manifest of the OSGi application that is contained in the EBA asset.
- The bundles and their versions that are listed in the deployment manifest must be available, either within the EBA file or from a bundle repository.
- If the asset has previously been updated, the bundle downloads for the previous update must have completed. See “Checking the bundle download status of an EBA asset” on page 1076.

You can import a deployment manifest by using the administrative console, as described in this topic, or by using wsadmin commands, as described in “Importing a deployment manifest using the importDeploymentManifest command” on page 1101.

About this task

A deployment manifest file, META-INF/DEPLOYMENT.MF, is created automatically when you import an EBA asset. The deployment manifest file lists, at specific versions, all the bundles and composite bundles that make up the application, including bundles that are determined following dependency analysis. The manifest file is used to ensure that each time an application server starts, the bundles that make up the application are the same.

You can export the deployment manifest file from an application, then import the manifest file into another instance of the same application located somewhere else. This is useful during application development, when an application is fully tested and moves to a production environment. By importing the deployment manifest from the test environment, you ensure that the bundles and their versions that make up the application in the production environment are exactly the same as the bundles that make up the application in the test environment.

When you import a deployment manifest into an EBA asset, the content of the deployment manifest must correspond with the existing application manifest in the EBA asset.

The application resolving process checks whether the deployment manifest contains all the required bundles. It should not need to pull in extra bundles to resolve the EBA asset.

Procedure

1. Start the administrative console.
2. Navigate to **Applications** > **Application Types** > **Assets**.
3. Click the name of the asset into which you want to import the deployment manifest. The Asset settings panel is displayed. If the bundle downloads for any previous update have completed, then the option to import a deployment manifest is displayed under the Additional Properties section.
4. Click **[Additional Properties] Import a deployment manifest into this application**.

   The Import a deployment manifest into this application panel is displayed.
5. Click **Local file system** or **Remote file system**, as required, then enter the fully-qualified path to the deployment manifest file that you want to import. The file must be a valid deployment manifest, with a `.MF` file extension.
6. Click **OK**. The deployment manifest is checked to ensure that the bundles that it lists can be resolved, then it is imported into the EBA asset. The file name changes to `DEPLOYMENT.MF`. Any new bundles that are required to provision the application are downloaded.
7. Save your changes to the master configuration.
8. Optional: Check the update status of the composition unit.

   If you plan to update the composition unit at this time, check the update status of the associated OSGi composition unit. This status is one of the following values:
   - Using latest OSGi application deployment.
   - New OSGi application deployment not yet available because it requires bundles that are still downloading.
   - New OSGi application deployment available.
   - New OSGi application deployment cannot be applied because bundle downloads have failed.
9. Optional: Update the composition unit to the latest deployment.

   When all bundle downloads are complete, you can update the OSGi composition unit so that the business-level application uses the newer configuration. You do not have to update the composition unit every time you update the asset. If any of the updates contain configuration options, you update the configuration information. You can also take the opportunity to make additional, non-essential configuration changes. When you save the changes to the composition unit, the associated business-level application is updated to use the new configuration. If the business-level application is running, the bundle and configuration updates are applied immediately. If possible (that is, depending on the nature of the updates) the system applies the updates without restarting the application. If new packages are imported, or new services injected, then the application is restarted. These packages and services can come from newly-provisioned bundles, or from bundles that have already been provisioned.

Importing a deployment manifest using the importDeploymentManifest command

You can use this command to import a suitable deployment manifest to an enterprise bundle archive (EBA) asset. When you import the file, the bundles are resolved. If the bundles cannot be resolved, the import does not complete and an exception message is generated.

Before you begin

The file to import must be a valid deployment manifest file, using the naming format *file_name*`.MF`, for example `DEPLOYMENT_TEST.MF`. When you import the deployment manifest into the EBA asset, the file is renamed to `DEPLOYMENT.MF`.

For the import to succeed, the following conditions must be met:

- The deployment manifest to import must correspond with the application manifest of the OSGi application that is contained in the EBA asset.
- The bundles and their versions that are listed in the deployment manifest must be available, either within the EBA file or from a bundle repository.

- If the asset has previously been updated, the bundle downloads for the previous update must have completed. See “Checking the bundle download status of an EBA asset” on page 1076.

You can import a deployment manifest by using the wsadmin commands, as described in this topic, or by using administrative console, as described in “Importing a deployment manifest” on page 1100.

About this task

A deployment manifest file, `META-INF/DEPLOYMENT.MF`, is created automatically when you import an EBA asset. The deployment manifest file lists, at specific versions, all the bundles and composite bundles that make up the application, including bundles that are determined following dependency analysis. The manifest file is used to ensure that each time an application server starts, the bundles that make up the application are the same.

You can export the deployment manifest file from an application, then import the manifest file into another instance of the same application located somewhere else. This is useful during application development, when an application is fully tested and moves to a production environment. By importing the deployment manifest from the test environment, you ensure that the bundles and their versions that make up the application in the production environment are exactly the same as the bundles that make up the application in the test environment.

When you import a deployment manifest into an EBA asset, the content of the deployment manifest must correspond with the existing application manifest in the EBA asset.

The application resolving process checks whether the deployment manifest contains all the required bundles. It should not need to pull in extra bundles to resolve the EBA asset.

Procedure

1. Start the wsadmin scripting client if it is not already running.
2. Use the importDeploymentManifest command. For example:

   ```
   AdminTask.importDeploymentManifest('[-asset
   com.ibm.ws.eba.example.blabber.app.eba -file /test/temp/DEPLOYMENT.MF]')
   ```

   The deployment manifest is checked to ensure that the bundles that it lists can be resolved.
3. Save your changes to the master configuration.

   To save your configuration changes, use the following command:

   ```
   AdminConfig.save()
   ```

   The deployment manifest is imported into the EBA asset and any new bundles that are required to provision the application are downloaded. The file name changes to `DEPLOYMENT.MF`.
4. Optional: Check the update status of the composition unit.

   If you plan to update the composition unit at this time, check the update status of the associated OSGi composition unit. This status is one of the following values:
   - Using latest OSGi application deployment.
   - New OSGi application deployment not yet available because it requires bundles that are still downloading.
   - New OSGi application deployment available.
   - New OSGi application deployment cannot be applied because bundle downloads have failed.
5. Optional: Update the composition unit to the latest deployment.

   When all bundle downloads are complete, you can update the OSGi composition unit so that the business-level application uses the newer configuration. You do not have to update the composition unit every time you update the asset. If any of the updates contain configuration options, you update

the configuration information. You can also take the opportunity to make additional, non-essential configuration changes. When you save the changes to the composition unit, the associated business-level application is updated to use the new configuration. If the business-level application is running, the bundle and configuration updates are applied immediately. If possible (that is, depending on the nature of the updates) the system applies the updates without restarting the application. If new packages are imported, or new services injected, then the application is restarted. These packages and services can come from newly-provisioned bundles, or from bundles that have already been provisioned.

importDeploymentManifest command:

Use the importDeploymentManifest command to import a deployment manifest to an enterprise bundle archive (EBA) asset. When you import the file, the bundles are resolved. If the bundles cannot be resolved, the import does not complete and an exception message is generated.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command imports a deployment manifest to an EBA asset.

For the import to succeed, the following conditions must be met:

- The deployment manifest to import must correspond with the application manifest of the OSGi application that is contained in the EBA asset.
- The bundles and their versions that are listed in the deployment manifest must be available, either within the EBA file or from a bundle repository.
- If the asset has previously been updated, the bundle downloads for the previous update must have completed. See “Checking the bundle download status of an EBA asset” on page 1076.

When the deployment manifest is successfully imported to the EBA asset, its file name changes to `DEPLOYMENT.MF`, and any new bundles that are required to provision the application are downloaded.

Target object

None.

Required parameters

`-asset`
The name of the EBA asset to import the deployment manifest into. This must be an installed EBA asset.

-file
The fully qualified file path for the location of the deployment manifest to import.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.importDeploymentManifest('[-asset
com.ibm.ws.eba.example.blabber.app.eba -file /test/temp/DEPLOYMENT.MF]')
```

Default messaging provider, JMS resources

WebSphere Application Server supports asynchronous messaging through the use of the Java Message Service (JMS).

The default messaging provider enables enterprise applications deployed on WebSphere Application Server to perform asynchronous messaging without the need for you to install a JMS provider. The default messaging provider is installed and runs as part of WebSphere Application Server.

The default messaging provider is based on service integration technologies. You can use the WebSphere Application Server administrative console to configure JMS resources:

- *JMS connection factories* that applications use to connect to a service integration bus.
- *JMS queues* that applications use to send messages to and receive messages from. An application sends messages to a specific queue and those messages are retrieved and processed by another application listening to that queue. JMS queues are assigned to queue destinations on a service integration bus. Such JMS queues are available, over a long period of time, to all applications with access to the bus.
- *JMS topics* that applications can use as named collection points for messages. To send messages, applications publish messages to topics. To receive messages, applications subscribe to topics. JMS topics are assigned to topic spaces on the bus. Such JMS topics are available, over a long period of time, to all applications with access to the bus.

Chapter 20. OSGiApplicationCommands: OSGi Applications administrative commands for the AdminTask object

You can use these administrative commands to manage your OSGi applications.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided so that you can make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

addExternalBundleRepository command

Use the addExternalBundleRepository command to add a link to an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a link to an external bundle repository.

Target object

None

Required parameters

-name *bundle_repository_name*
The name by which you want the external bundle repository link to be known.

-url *bundle_repository_URL*
The URL of the bundle repository XML file. For example, `http://external_location/repository.xml`.

Conditional parameters

None.

Optional parameters

-description *bundle_repository_description*
An optional description of the bundle repository.

Example

```
AdminTask.addExternalBundleRepository('-name bundle_repository_name
                                -url http://external_location/repository.xml
                               [-description bundle_repository_description]')
```

Square brackets ("[ ]") indicate that a parameter is optional.

addLocalRepositoryBundle command

Use the addLocalRepositoryBundle command to add a bundle, composite bundle or grouped-up set of bundles to the internal bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a bundle, a composite bundle or a grouped-up set of bundles to the internal bundle repository.

You can install bundles singly, or you can install a set of bundles packaged as a compressed archive file with a `.zip` file extension. In both cases, the bundles are available individually in the repository. If you install a composite bundle in a bundle repository, and the composite bundle includes bundles by reference, you must ensure that the referenced bundles are also available in the same repository. If you use the internal bundle repository, and the composite bundle directly contains bundles, the contained bundles are not listed separately and are only available as part of the composite bundle. For more information, see Composite bundles.

Target object

None

Required parameters

-file *path*
The path and file name of a compressed archive file that has a `.jar`, `.cba` or `.zip` file extension and is available on the server file system.

Each individual bundle must be packaged as a `.jar` file, and must contain a suitably-configured bundle manifest file. Each composite bundle must be packaged as a compressed archive file with a `.cba` file extension, and must contain a suitably-configured composite bundle manifest file. Each grouped-up set of bundles must be packaged as a compressed archive file with a `.zip` file extension.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.addLocalRepositoryBundle('-file path')
```

addOSGiExtension command

Use the addOSGiExtension command to add a composite bundle as an extension to a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a composite bundle as an extension to a composition unit. The composite bundle must be available in the internal bundle repository, or in an external repository that can process composite bundles.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-symbolicName *cba_symbolic_name*
The non-localizable name for this composite bundle.

-version *cba_version*
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`. The symbolic name, together with the version, identifies a unique composite bundle.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.addOSGiExtension('
  -cuName cu_name
  -symbolicName cba_symbolic_name
  -version cba_version
')
```

addOSGiExtensions command

Use the addOSGiExtensions command to add several composite bundles as extensions to a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds several composite bundles as extensions to a composition unit. The composite bundles must be available in the internal bundle repository, or in an external repository that can process composite bundles.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-extensions

A list of the composite bundle extensions to be added. Each list entry contains the symbolic name and the version for a composite bundle. The symbolic name, together with the version, identifies a unique composite bundle.

cba_symbolic_name
The non-localizable name for this composite bundle.

cba_version
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.addOSGiExtensions([
  '-cuName', 'cu_name',
  '-extensions',
    'cba1_symbolic_name;cba1_version
     cba2_symbolic_name;cba2_version
     cba3_symbolic_name;cba3_version
    '
])
```

exportDeploymentManifest command

Use the exportDeploymentManifest command to export the current deployment manifest, the `DEPLOYMENT.MF` file, from an enterprise bundle archive (EBA) asset. You might want to do this to save the information, or to import it into another identical application.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

This command exports the current deployment manifest from an EBA asset.

Target object

None.

Required parameters

-asset
The name of the EBA asset to export the deployment manifest from. This must be an installed EBA asset.

-path
The file path for the location of the exported deployment manifest. If the location does not exist, the export process creates it.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.exportDeploymentManifest('[-asset
com.ibm.ws.eba.example.blabber.app.eba -path /test/temp/]')
```

importDeploymentManifest command

Use the importDeploymentManifest command to import a deployment manifest to an enterprise bundle archive (EBA) asset. When you import the file, the bundles are resolved. If the bundles cannot be resolved, the import does not complete and an exception message is generated.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command imports a deployment manifest to an EBA asset.

For the import to succeed, the following conditions must be met:

- The deployment manifest to import must correspond with the application manifest of the OSGi application that is contained in the EBA asset.
- The bundles and their versions that are listed in the deployment manifest must be available, either within the EBA file or from a bundle repository.
- If the asset has previously been updated, the bundle downloads for the previous update must have completed. See “Checking the bundle download status of an EBA asset” on page 1076.

When the deployment manifest is successfully imported to the EBA asset, its file name changes to `DEPLOYMENT.MF`, and any new bundles that are required to provision the application are downloaded.

Target object

None.

Required parameters

-asset
The name of the EBA asset to import the deployment manifest into. This must be an installed EBA asset.

-file
The fully qualified file path for the location of the deployment manifest to import.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.importDeploymentManifest('[-asset
com.ibm.ws.eba.example.blabber.app.eba -file /test/temp/DEPLOYMENT.MF]')
```

listExternalBundleRepositories command

Use the listExternalBundleRepositories command to list all links to external bundle repositories.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

This command lists all links to external bundle repositories. The list includes any repository links that you have added since you last saved your changes, and excludes any repository links that you have removed since you last saved your changes.

Target object

None.

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listExternalBundleRepositories()
```

listLocalRepositoryBundles command

Use the listLocalRepositoryBundles command to list all bundles held in the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all bundles that are held in the internal bundle repository. The list includes any bundles that you have added since you last saved your changes, and excludes any bundles that you have removed since you last saved your changes.

Target object

None.

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listLocalRepositoryBundles()
```

listOSGiExtensions command

Use the listOSGiExtensions command to list the symbolic names and versions of all the extensions that are currently added to a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists the symbolic names and versions of all the extensions that are currently added to a composition unit. The output from the listOSGiExtensions command is formatted so that you can copy the list of extensions, then paste them into the removeOSGiExtensions command.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listOSGiExtensions('-cuName cu_name')
```

modifyExternalBundleRepository command

Use the modifyExternalBundleRepository command to modify a link to an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command modifies a link to an external bundle repository.

Target object

The specified external bundle repository link.

Required parameters

-name *bundle_repository_name*
The name of the external bundle repository link. You cannot change this value. You can use the listExternalBundleRepositories command to list the names of existing bundle repository links.

Conditional parameters

None.

Optional parameters

-url *bundle_repository_URL*
The modified URL for the external bundle repository XML file.

-description *bundle_repository_description*
The modified description of the external bundle repository link.

Example

```
AdminTask.modifyExternalBundleRepository('-name bundle_repository_name
                              [-url http://external_location/repository.xml]
                              [-description bundle_repository_description]')
```

Square brackets (“[ ]”) indicate that a parameter is optional.

removeExternalBundleRepository command

Use the removeExternalBundleRepository command to remove a link to an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('OSGiApplicationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes a link to an external bundle repository.

Target object

The specified bundle repository link.

Required parameters

-name *bundle_repository_name*
The name by which the bundle repository link is known. You can use the listExternalBundleRepositories command to list the names of existing bundle repository links.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeExternalBundleRepository('-name bundle_repository_name')
```

removeLocalRepositoryBundle command

Use the removeLocalRepositoryBundle command to remove a bundle or composite bundle from the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes a bundle or composite bundle from the internal bundle repository.

You cannot delete a bundle if it is contained in, or referenced by, a composite bundle. You must delete the composite bundle first, then delete the bundle.

Target object

The specified bundle.

Required parameters

-symbolicName *bundle_symbolic_name*
One of the following:

- The non-localizable name for this bundle.
- The bundle symbolic name of this composite bundle.

The bundle symbolic name, together with the bundle version, identifies a unique bundle.

-version *bundle_version*
One of the following:

- The version of this bundle.
- The version of this composite bundle.

The bundle version is in the form *n.n.n*, for example `1.1.0`. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

Note: You can use the listLocalRepositoryBundles command to list the symbolic names and versions of the bundles currently held in the repository.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeLocalRepositoryBundle('-symbolicName bundle_symbolic_name
                                       -version bundle_version')
```

removeLocalRepositoryBundles command

Use the removeLocalRepositoryBundles command to remove bundles and composite bundles from the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes bundles and composite bundles from the internal bundle repository.

You cannot remove composite bundles at the same time as you remove any bundles that they reference. You must first remove the composite bundles, then run the removeLocalRepositoryBundles command a second time to remove the referenced bundles.

You can use the listLocalRepositoryBundles command to list the symbolic names and versions of the bundles currently held in the repository. The output from the listLocalRepositoryBundles command is formatted so that you can copy the list of bundles, then paste them into the removeLocalRepositoryBundles command.

Target object

The specified bundles.

Required parameters

A list of the bundles to be removed.

The list entry for each bundle or composite bundle contains the bundle symbolic name and the bundle version. The bundle version is in the form *n.n.n*, for example `1.1.0`. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeLocalRepositoryBundles([
    'bundle1_symbolic_name;bundle1_version
     bundle2_symbolic_name;bundle2_version
     bundle3_symbolic_name;bundle3_version
    '
])
```

removeOSGiExtension command

Use the removeOSGiExtension command to remove a composite bundle extension from a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes a composite bundle extension from a composition unit.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-symbolicName *cba_symbolic_name*
The non-localizable name for this composite bundle.

-version *cba_version*
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`. The symbolic name, together with the version, identifies a unique composite bundle.

Note: You can use the listOSGiExtensions command to list the symbolic names and versions of all the extensions that are currently added to a composition unit.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeOSGiExtension('
  -cuName cu_name
  -symbolicName cba_symbolic_name
  -version cba_version
')
```

removeOSGiExtensions command

Use the removeOSGiExtensions command to remove several composite bundle extensions from a composition unit.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes several composite bundle extensions from a composition unit.

You can use the listOSGiExtensions command to list the symbolic names and versions of all the extensions that are currently added to a composition unit. The output from the listOSGiExtensions command is formatted so that you can copy the list of extensions, then paste them into the removeOSGiExtensions command.

Target object

The specified composition unit.

Required parameters

-cuName *cu_name*
The name of the composition unit.

-extensions

A list of the composite bundle extensions to be removed. Each list entry contains the symbolic name and the version for a composite bundle. The symbolic name, together with the version, identifies a unique composite bundle.

cba_symbolic_name
The non-localizable name for this composite bundle.

cba_version
The version of this composite bundle.

The composite bundle version is in the form *n.n.n*, for example `1.1.0`.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.removeOSGiExtensions([
  '-cuName', 'cu_name',
  '-extensions',
    'cba1_symbolic_name;cba1_version
     cba2_symbolic_name;cba2_version
     cba3_symbolic_name;cba3_version
    '
])
```

showExternalBundleRepository command

Use the showExternalBundleRepository command to show the configured parameters of an external bundle repository.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command shows the name, description when available, and URL of the bundle repository XML file, of an external bundle repository.

Target object

None.

Required parameters

-name *repository_name*
The name of a link to an external bundle repository.

Note: You can use the listExternalBundleRepositories command to list all links to external bundle repositories.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.showExternalBundleRepository(-name ExternalRepository1)
```

showLocalRepositoryBundle command

Use the showLocalRepositoryBundle command to show further details (in particular all manifest header entries) for a bundle in the bundle repository that is included in the product.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts.

Command-line help is provided for OSGi Applications commands:

- For a list of the available OSGi Applications commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('OSGiApplicationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command shows the values of the following properties for the specified bundle stored in the internal bundle repository:

- Bundle symbolic name
- Bundle version
- Bundle name
- Bundle description
- Imported packages
- Exported packages
- Required bundles

For more information about these properties, see Internal bundle repository [Settings].

Target object

The specified bundle.

Required parameters

-symbolicName *bundle_symbolic_name*
The non-localizable name for this bundle. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

-version *bundle_version*
The version of this bundle. The bundle version is in the form *n.n.n*, for example `1.1.0`. The bundle symbolic name, together with the bundle version, identifies a unique bundle.

Note: You can use the listLocalRepositoryBundles command to list the symbolic names and versions of the bundles currently held in the repository.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.showLocalRepositoryBundle('-symbolicName bundle_symbolic_name
                                    -version bundle_version')
```

Chapter 21. Administering Portlet applications

This page provides a starting point for finding information about portlet applications, which are special reusable Java servlets that appear as defined regions on portal pages. Portlets provide access to many different applications, services, and web content.

Portlet container settings and custom properties

Portlet container settings

Use this page to configure and manage the portlet container of this application server.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Portlet Container Settings** > **Portlet container**.

Enable portlet fragment cache

Specifies whether to create a cached entry when a portlet is invoked, similar to servlet caching of the web container settings.

Portlet fragment caching requires that servlet caching is enabled. Therefore, enabling portlet fragment caching automatically enables servlet caching. Disabling servlet caching automatically disables portlet fragment caching.

Portlet container custom properties

You can configure name-value pairs of data, where the name is a property key and the value is a string value that you can use to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console. The following is a list of the available Portlet container custom properties.

To specify Portlet container custom properties:

1. In the administrative console click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Portlet Container Settings** > **Portlet container**.
2. Under **Additional Properties** select **Custom Properties**.
3. On the Custom Properties page, click **New**.
4. On the settings page, enter the name of the custom property that you want to configure in the **Name** field and the value that you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

You can use the custom properties page to define the following portlet container custom properties:

- "useShortMBeanNames"

useShortMBeanNames

Portlet MBeans are registered by both their short name and full name. To enable the use of short MBean names for Portlet and PortletApplication MBeans, create the following name-value pair:

Name	Value
useShortMBeanNames	true

The default is false.

MBeans registered by the full identifiable name, have the following format:

```
<ApplicationName>#<WARfilename.war>_portlet.<portlet_name> for the Portlet MBean
<ApplicationName>#<WARfilename.war>_portlet for the PortletApplication MBean
```

where <..> is replaced by the corresponding application data. For example, `SampleApplication#SamplePortlet.war_portlet.SamplePortlet`.

Portlet and PortletApplication MBeans

The MBeans of type portlet and portletapplication provide information about a given portlet application and its portlets. Through the MBean of type portletapplication, you can retrieve a list of names of all portlets that belong to a portlet application. By querying the MBean of type portlet with a given portlet name, you can retrieve portlet specific information from the MBean of type portlet.

Each MBean that corresponds to a portlet or portlet application is uniquely identifiable by its name. Portlet applications are not required to have a name set within the `portlet.xml`. The MBean name for MBeans of the portletapplication type is the enterprise archive (EAR) file name followed by "#" and the web module name concatenated with the string "_portlet". For example, portletapplication type MBeans have the following format:

```
<EarFileName>#<WarFileName>_portlet
```

The name chosen for the MBean of type portlet is the name of the MBean of type portletapplication that the portlet belongs to, concatenated with the portlet name:

```
<EarFileName>#<WarFileName>_portlet.<portletname>
```

The following is an example of the resulting PortletApplication MBean name and portlet names:

```
EarName             SampleEar
WebModule             SampleWar.war

PortletApplication MBean name:   SampleEar#SampleWar_portlet
Portlet:             SampleEar#SampleWar_portlet.BookmarkPortlet
```

The MBean names have been changed compared to version 6.1, because the old naming patterns are not unique and can lead to problems under certain circumstances. If you rely on the old naming pattern, you can set the portlet container custom property, `useShortMBeanNames`, to true to activate the previous known MBean names. Because this is a performance impact, you might not want to activate the old naming pattern if it is not necessary.

A full stop separates the preceding web module name from the portlet name. Review the Portlet and PortletApplication MBean type API documentation for additional information. The generated API documentation is available in the information center table of contents from the path, **Reference** > **Administrator** > **API documentation** > **MBean interfaces**.

The following code is an example of how to invoke the MBean of type portletapplication for an application with the name, SampleWar.

```
String myPortletApplicationName = "SampleEar#SampleWar_portlet";
This name is composed by the Ear file name followed by "#" and
the web module name concatenated with the substring "_portlet"


com.ibm.websphere.management.AdminService adminService =
  com.ibm.websphere.management.AdminServiceFactory.getAdminService();
javax.management.ObjectName on =
  new ObjectName("WebSphere:type=PortletApplication,name=" + myPortletApplicationName + ",*");

Iterator onIter = adminService.queryNames(on, null).iterator();
```

```
while(onIter.hasNext())
{
 on = (ObjectName)onIter.next();
}

String ctxRoot = (java.lang.String)adminService.getAttribute(on, "webApplicationContextRoot");
```

In the previous example, the MBeanServer is first queried for an MBean of type portletapplication. If this query is successful, the webApplicationContextRoot attribute is retrieved on that MBean or the first MBean that is found. The result is stored in the ctxRoot variable. This variable now contains the context root of the web application that contains the portlet application that was searched. The variable is similar to "/bookmark".

The next code example demonstrates how to invoke the MBean of type portlet for a portlet with the name, BookmarkPortlet.

```
String myPortletName = "SampleEar#SampleWar_portlet.BookmarkPortlet";
This name is composed by the name of the MBean of type portletapplication and
the portlet name, separated by a full stop because the same portlet name may
be used within different web modules, but must be unique within the system.


com.ibm.websphere.management.AdminService adminService =
  com.ibm.websphere.management.AdminServiceFactory.getAdminService();
javax.management.ObjectName on =
  new ObjectName("WebSphere:type=Portlet,name=" + myPortletName + ",*");
Iterator iter = adminService.queryNames(on, null).iterator();

while(iter.hasNext())
{
 on = (ObjectName)iter.next;
}

java.util.Locale locale = (java.util.Locale) adminService.getAttribute(on, "defaultLocale");
```

The locale returned by the method getAttribute method for the MBean is the default locale defined for this portlet.

Full names for Portlet and PortletApplication MBeans

MBeans are also registered by the full identifiable name:

```
<ApplicationName>#<WARfilename.war>_portlet.<portlet_name> for the Portlet MBean
<ApplicationName>#<WARfilename.war>_portlet for the PortletApplication MBean
```

where <..> is replaced by the corresponding application data. For example, `SampleApplication#SamplePortlet.war_portlet.SamplePortlet`. You can enable the short MBean names by setting the `useShortMBeanNames` portlet container custom property to true.

Chapter 22. Administering Scheduler service

This page provides a starting point for finding information about the scheduler service, a WebSphere programming extension responsible for starting actions at specific times or intervals.

Schedulers are persistent and transactional timer services that run Enterprise JavaBeans methods or send Java Message Service messages using any Java Message Service messages using any Java Platform, Enterprise Edition (Java EE) server application.

The scheduler service helps minimize IT costs and increase application speed and responsiveness by maximizing utilization of existing computing resources.

The scheduler service provides the ability to reliably process workloads using parallel processing and schedule resource-intensive tasks to process during low traffic off-hours.

Installing default scheduler calendars

Scheduler calendars

The scheduler provides stateless session bean interfaces which allow creating common calendars which can be used by the scheduler and any Java Platform, Enterprise Edition (Java EE) application.

The `SchedulerCalendars.ear` application is available and provides a default UserCalendar Enterprise Java Beans (EJB) implementation which allows using the SIMPLE and CRON calendars. Although this application is not required when using the scheduler, it is available to use from any Java EE application.

For details on how the SIMPLE and CRON calendars behave, see the API documentation for the com.ibm.websphere.scheduler.UserCalendar interface.

Specifying a UserCalendar with the scheduler

A UserCalendar is specified using the setUserCalendar() method of the TaskInfo interface of the scheduler. This interface allows you to select the Java Naming and Directory Interface (JNDI) name of the home interface of a UserCalendar bean. Because some UserCalendar bean implementations might handle multiple types of calendars, the interface also allows you to optionally select which type of calendar to use. A list of valid calendar types can be retrieved by invoking the getCalendarNames() method of the UserCalendar interface.

If the setUserCalendar() method is not invoked, or if a value of null or empty-string is specified for the home JNDI name parameter, then the default UserCalendar is used internally by the scheduler. When the default UserCalendar is accessed internally, it is not necessary that the `SchedulerCalendars.ear` system application be installed.If you want to use the default UserCalendar with a CRON entry, you must switch to the CRON entry manually. The following code sample shows this switch:

```
BeanTaskInfo taskInfo = (BeanTaskInfo)scheduler.createTaskInfo(BeanTaskInfo.class);
String calendarVariant = "CRON";
taskInfo.setUserCalendar(null, calendarVariant);
// cron table entry
String cronTableEntry = "0 17,20,23 * ? * *";
taskInfo.setStartTimeInterval(cronTableEntry);
```

You might want to use the default UserCalendar directly in your other Java EE applications, apart from the scheduler. In this case, you may use the UserCalendarHome.DEFAULT_CALENDAR_JNDI_NAME value to look up the default UserCalendar from your applications. You may also supply this value to the setUserCalendar() method of the TaskInfo interface. You will need to ensure the `SchedulerCalendars.ear` system application was either automatically installed or that you have installed it manually.

Installing default scheduler calendars

The default scheduler SIMPLE and CRON calendars are available in the `SchedulerCalendars.ear` system application and are automatically installed on standalone server profiles. System applications cannot be installed and uninstalled like traditional Java Platform, Enterprise Edition (Java EE) applications.

About this task

The following steps are required to map the `SchedulerCalendars.ear` system application on a server or cluster in a network deployment environment.

Procedure

1. Start the wsadmin tool and connect to the deployment manager.
2. Install the system application.
   - To install on a non-clustered server:
     - Using Jacl:

       ```
       $AdminApp install
       "/${WAS_INSTALL_ROOT}/systemApps/SchedulerCalendars.
       ear" {-systemApp -appname SchedulerCalendars -cell
        mycell -node mynode -server myserver}
       ```

     - Using Jython list:

       ```
       AdminApp.install('${WAS_INSTALL_ROOT}/systemApps/
       SchedulerCalendars.ear', ['-systemApp', '–appname',
        'SchedulerCalendars','-cell', 'mycell', '-node',
        'mynode', '-server', 'myserver'])
       ```

     - Using Jython string:

       ```
       AdminApp.install('${WAS_INSTALL_ROOT}/systemApps/
       SchedulerCalendars.ear', '[-systemApp –appname
        SchedulerCalendars –cell mycell –node mynode
        -server myserver]')
       ```

       Where:

Table 69. Variable options.. Variable values and options

Value	Option
mycell	the value of the cell option
mynode	the value of the node option
myserver	the value of the server option

   - To install on a cluster:
     - Using Jacl:

       ```
       $AdminApp install
       "\${WAS_INSTALL_ROOT}/systemApps/SchedulerCalendars.
       ear" {-systemApp -appname SchedulerCalendars -cell
        mycell -cluster mycluster}
       ```

     - Using Jython list:

       ```
       AdminApp.install('${WAS_INSTALL_ROOT}/systemApps/
       SchedulerCalendars.ear', ['-systemApp', '–appname',
        'SchedulerCalendars','-cell', 'mycell', '-cluster',
        'mycluster'])
       ```

     - Using Jython string:

       ```
       AdminApp.install('${WAS_INSTALL_ROOT}/systemApps/
       SchedulerCalendars.ear', '[-systemApp –appname
        SchedulerCalendars –cell mycell -cluster mycluster]')
       ```

       Where:

Table 70. Variable options.. Variable values and options

Value	Option
mycell	the value of the cell option
mycluster	the value of the cluster option

3. Save the configuration changes. Refer to the Saving configuration changes with the wsadmin tool topic for more information.

Uninstalling default scheduler calendars

The default scheduler SIMPLE and CRON calendars are available in the `SchedulerCalendars.ear` system application and are automatically installed on standalone server profiles. System applications cannot be installed and uninstalled like traditional Java Platform, Enterprise Edition (Java EE) applications.

About this task

Remove the SchedulerCalendars system application on federated node, as follows:

Procedure

1. Open a command window on the federated node.
2. Run the following command:

   On Unix platforms:

   ```
   $Install_Root/bin/wsadmin.sh -conntype none -profile $Profile_Name
   ```

   On Windows platforms:

   ```
   $Install_Root/bin/wsadmin -conntype none -profile $Profile_Name
   ```

   where:

   - *Install_Root* is the directory where WebSphere Application Server is installed.
   - *Profile_Name* is the name of the profile where the target server is located.
3. At the **wsadmin>** prompt, enter the following command for each server that exists on the node where you want to have the SchedulerCalendars application available:

   ```
   wsadmin> $AdminApp uninstall SchedulerCalendars "-cell $MyCell -node $MyNode -server $MyServer"
   ```

   where:

   - *$Install_Root* is the directory where WebSphere Application Server is installed.
   - *$MyCell*, *$MyNode*, and *$MyServer* are the values with the name of the cell, node, and server.

   Important: Each of these values are case-sensitive.
4. Repeat step three for each server in the current profile for which you will uninstall the SchedulerCalendars application.
5. When uninstallation is complete for the system application on all appropriate servers, enter the following commands:

   ```
   wsadmin> $AdminConfig save
   wsadmin> exit
   ```
6. Using the Administrative Console or scripting, start or restart the servers to unload the SchedulerCalendars application.

Results

The SchedulerCalendars application should now be removed.

Example: Using default scheduler calendars

The SIMPLE and CRON calendars can be used from any J2EE application. This topic describes that process.

Using default scheduler calendars involves looking-up the default UserCalendarHome Enterprise JavaBeans (EJB) home object, creating the UserCalendar bean and calling the applyDelta() method. For details on the applyDelta method as well as the syntax for the SIMPLE and CRON calendars, see the UserCalendar interface topic.

Example:

```
import java.util.Date;
import javax.naming.InitialContext;
import javax.rmi.PortableRemoteObject;
import com.ibm.websphere.scheduler.UserCalendar;
import com.ibm.websphere.scheduler.UserCalendarHome;

// Create an initial context
InitialContext ctx = new InitialContext();

// Lookup and narrow the default UserCalendar home.
UserCalendarHome defaultCalHome=(UserCalendarHome)
 PortableRemoteObject.narrow(ctx.lookup(
      UserCalendarHome.DEFAULT_CALENDAR_JNDI_NAME),
    UserCalendarHome.class);

// Create the default UserCalendar instance.
UserCalendar defaultCal = defaultCalHome.create();

// Calculate a date using CRON based on the current
// date and time.  Return the next date that is
// Saturday at 2AM
Date newDate =
    defaultCal.applyDelta(new Date(),
        "CRON", "0 0 2 ? * SAT");
```

Managing schedulers

Managing schedulers

Schedulers are configured using the administrative console, configuration service or scripting and are available to all servers on which a scheduler is visible.

About this task

You can create multiple schedulers within a single server, cluster, node or cell. Each configured scheduler is an independent task scheduling engine that has a unique Java Naming and Directory Interface (JNDI) name, persistent storage device and daemon.

Procedure

1. Configure schedulers.
2. Create the database for schedulers.

Scheduler daemon

A scheduler daemon is a background thread that searches for tasks to run in the database.

A scheduler daemon is started for each scheduler defined on each server. If `Scheduler 1` is configured on `server1`, then only one scheduler daemon runs on `server1` unless it is cloned. If `Scheduler 1` is defined at the node scope level, then the scheduler will run on each server within that node.

The poll interval determines the frequency at which the persistent store is queried. By default, this value is set to 30 seconds. When a task is found that is scheduled to run within the current poll interval, an asynchronous beans alarm is set. The task then runs as close to this time as possible using an alarm thread from the scheduler's associated work manager. Thus, the number of alarm threads configured on the work manager determines how many concurrent tasks are executed. No tasks are lost. If we reach this limit, then new tasks are simply queued to be executed when an alarm thread becomes available. The actual firing time is dictated by server load and availability of free threads in the alarm thread pool of the associated work manager.

Scheduler daemons in a cluster

When multiple schedulers are configured to use the same tables (as is the case in a clustered environment), any of the daemons can find a task and set the alarm in its Java virtual machine (JVM). The task is executed in the virtual machine where the scheduler daemon first runs, until the daemon is stopped and another daemon starts. If an application on `server1` schedules a task to run and `server2` was started before `server1`, then the task runs on `server2`.

Example: Stopping and starting scheduler daemons using Java Management Extensions API

Use the wsadmin scripting tool to invoke a `Jacl` script and stop and start a scheduler daemon.

This example JACL script can be invoked using the wsadmin scripting tool. It will attempt to stop and start a scheduler daemon.

```
# Example JACL Script to restart a Scheduler Daemon

set schedJNDIName sched/MyScheduler

# Find the WASScheduler MBean
regsub -all {/} $schedJNDIName "." schedJNDIName
set mbeanName Scheduler_$schedJNDIName
puts "Looking up Scheduler MBean $mbeanName"
set sched [$AdminControl queryNames WebSphere:*,type=WASScheduler,name=$mbeanName]

# Invoke the stopDaemon operation.
puts "Stopping the daemon..."
$AdminControl invoke $sched stopDaemon
puts "The daemon has stopped."

# Invoke the startDaemon operation.
puts "Starting the daemon..."
$AdminControl invoke $sched startDaemon 0
puts "The daemon has started."
```

Example: Dynamically changing scheduler daemon poll intervals using Java Management Extensions API

Use the wsadmin scripting tool to invoke a `Jacl` script and dynamically change scheduler daemon poll intervals.

To dynamically change scheduler daemon poll intervals, use the wsadmin scripting tool to invoke this example `JACL` script. Invoking this example sets the poll interval of the scheduler daemon to 60 seconds.

```
# Example JACL Script to set the Scheduler daemon's poll interval

set schedJNDIName sched/MyScheduler

# Find the WASScheduler MBean
regsub -all {/} $schedJNDIName "." schedJNDIName
set mbeanName Scheduler_$schedJNDIName
puts "Looking-up Scheduler MBean $mbeanName"
```

```
set sched [$AdminControl queryNames WebSphere:*,type=WASScheduler,name=$mbeanName]

# Set the poll interval to 60 seconds (60000 ms)
$AdminControl setAttribute $sched pollInterval 60000
puts "Poll interval set."
```

Configuring schedulers

Before your application can make use of the scheduler service, you must configure a scheduler using the administrative console, configuration service or scripting. Conceptually, a scheduler is similar to a data source in that you must specify various configuration attributes, including a JNDI name where the instance is bound. Once defined, an application using the Scheduler API or WASScheduler MBean can look up the scheduler object and call various methods to manage tasks.

About this task

The scheduler service is always enabled. In previous versions of the product, the scheduler service could be disabled using the administrative console or configuration service. Scheduler service configuration objects are present in the configuration service, but the enabled attribute is ignored.

To achieve high availability, you can configure a duplicate scheduler on each server in a cluster, or create a scheduler at the cluster scope. For example, each server that contains a scheduler with the JNDI name `sched/MyScheduler`, with the same database configuration parameters (data source and table prefix) behaves as a single clustered scheduler. Each server in the scheduler cluster has a running scheduler instance, which increases the number of poll daemons and allows automatic failover. For more information on creating clusters for high availability, see the article, "WebSphere Enterprise Scheduler planning and administration guide."

Typically, create schedulers at the server or cluster scope. Scheduler poll daemons run in each server within the configured scope, which means that if you create a scheduler at the node or cell scope, the scheduler poll daemon can attempt to run tasks on any of the servers in the node or cell. If applications are not mapped uniformly over each server in that scope, the scheduler might not run tasks correctly. Because applications are mapped to servers and clusters, there is less chance for error and less competition between daemons to run tasks.

Depending on your preferred method of configuration, select one of the following steps to configure schedulers.

Procedure

1. Configuring schedulers using the administrative console.
2. Configuring schedulers using Java Management Extensions API (JMX).

Results

A scheduler is configured and ready to use.

Configuring scheduler default transaction isolation

The scheduler uses read-committed transaction isolation, by default, when reading tasks using the get or find APIs on the com.ibm.websphere.scheduler.Scheduler interface and WASScheduler MBean. The default behavior for a scheduler can be changed to read-uncommitted, which allows the get and find methods to return the current or next state of the task in the database. This topic describes how to change the default behavior for the get and find methods.

About this task

See the scheduler API documentation to view the com.ibm.websphere.scheduler.TaskInfo.setTaskExecutionOptions() method, which details how to return the next state of the task or the current state of the task.

Attention: If the scheduler database does not support uncommitted reads, such as Oracle, this parameter has no effect.

To change the default behavior for the get and find methods, complete the following steps:

Procedure

1. From the administrative console, click **Resources** > **Schedulers** > *scheduler_name*.
2. Click **Custom Properties**.
3. Click **New**.
4. Add the following properties:

Name	defaultReadTransactionIso
Type	java.lang.Integer
Value	`1 (for read-uncommitted transaction isolation)` `2 (for read-committed transaction isolation)`

5. Click **Apply** or
6. Click **OK**.
7. Save the changes and verify that you initiate a file synchronization before restarting the servers.
8. Restart the application server for the changes to take effect.

Configuring schedulers using the administrative console

Schedulers can be created or configured using the administrative console.

Procedure

1. Start the administrative console.
2. Select **Resources** > **Schedulers**.
3. Click **New**.
4. Specify configuration settings.

   Fields marked with an asterisk (*) are required. The settings are described in detail in the topic "Scheduler settings".

   bprac: If you do not require interoperability with PME Version 5.0 tasks, under Additional Properties, click **Custom Properties > New**. Then specify `disableV50TaskInteroperability` in the **Name** field, `true` in the **Value** field, and select `java.lang.boolean` from the list of available Type options. When this property is set to `true`, tablespace scans that are only required for PME Version 5.0 tasks no longer occur.
5. Click **OK** or **Apply** to save the changes.
6. Save the changes to the configuration repository.

Results

A scheduler is now configured and ready to use for newly installed applications. If the scheduler JNDI name is not yet visible to your application, restarting the application or restarting application server will allow the scheduler to be seen.

When schedulers are created for the first time, the poll daemon will not automatically start and must be started manually and will only start automatically the next time the server is started. To start the poll daemon manually, refer to the scheduler daemons topic.

Attention: Changes to existing scheduler configurations will not take affect until after the application server is restarted.

Schedulers collection:

Use this page to manage scheduler configurations. Schedulers are persistent and transactional timer services that can run business logic. Each scheduler runs tasks independently and has a programming interface accessible from Java Platform, Enterprise Edition (Java EE) applications using the Java Naming and Directory Interface (JNDI). You can also manage schedulers using a Java Management Extensions (JMX) MBean. See the scheduler documentation in the Information Center for details on how to configure and use schedulers.

To view this administrative console page, click **Resources > Schedulers**.

Name:

Specifies the name of the data source where persistent tasks are stored.

Data type String

JNDI name:

Specifies the JNDI name of the work manager, which is used to manage the number of tasks that can run concurrently with the scheduler. The work manager also can limit the amount of Java EE context applied to the task.

The JNDI name specifies where this scheduler instance is bound in the name space. Clients can look this name up directly, although the use of resource references is recommended.

Data type String

Scope:

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Data source JNDI name:

Specifies the alias for the user name and password that are used to access the data source.

Any data source available in the name space can be used with a scheduler. Multiple schedulers can share a single data source while using different tables by specifying a table prefix.

Data type String

Table prefix:

Specifies the string prefix to affix to the scheduler tables.

Multiple independent schedulers can share the same database if each instance specifies a different prefix string.

Data type	String

Poll interval:

Specifies the interval, in seconds, that a scheduler polls the database. The default value is appropriate for most applications.

Each poll operation can be consuming. If the interval is extremely small and there are many scheduled tasks, polling can consume a large portion of system resources.

Data type	Integer
Units	Seconds
Default	30
Range	Any positive long integer

Work manager JNDI name:

Specifies the JNDI name of the work manager, which is used to manage the number of tasks that can run concurrently with the scheduler. The work manager also can limit the amount of Java EE context applied to the task.

The work manager is a server object that serves as a logical thread pool for the scheduler. Each repeating task that is created using this scheduler uses the **Number of alarm threads** specified in the work manager, which affects the number of tasks that can run concurrently. Use the work manager **Service Names** property to limit the amount of context information that is propagated to the task when it runs.

When a task runs, the task is run in the work manager associated with the scheduler instance. You can control the number of actively running tasks at a given time by configuring schedulers with a specific work manager. The number of tasks that can run concurrently is governed by the **Number of alarm threads** parameter on the work manager.

Verify tables:

Specifies to validate that scheduler data sources, table prefixes, security authentication information and tables are configured correctly.

You can use this verification method in production and development environments without altering database properties.

Create tables:

Specifies to create the necessary tables and indices required for a scheduler to operate.

This method of creating scheduler tables is designed for simple topologies and development environments. Use the supplied scheduler data definition language files for advanced or production environments and for databases that do not support this feature.

Drop tables:

Specifies the removal of tables and indices required for schedulers to operate.

This method of removing scheduler tables and indices is recommended for development environments and does not delete previously scheduled tasks.

Schedulers settings:

Use this page to modify scheduler settings.

To view this administrative console page, click **Resources > Schedulers >** *scheduler_name*.

Scope:

Specifies the scope of the configured resource. This value indicates the location for the configuration file.

Name:

Specifies the name by which this scheduler is known for administrative purposes.

Data type String

JNDI name:

Specifies the name of the data source where persistent tasks are stored.

The Java™ Naming and Directory Interface (JNDI) name specifies where this scheduler instance is bound in the namespace. Clients can look this name up directly, although the use of resource references is recommended.

Data type String

Description:

Specifies the description of this scheduler for administrative purposes.

Data type String

Category:

Specifies a string that can be used to classify or group this scheduler.

Data type String

Data source JNDI name:

Specifies the name of the data source where persistent tasks are stored.

Any data source available in the name space can be used with a scheduler. Multiple schedulers can share a single data source while using different tables by specifying a table prefix.

Data type String

Data source alias:

Specifies the alias for the user name and password that are used to access the data source.

Data type String

Table prefix:

Specifies the string prefix to affix to the scheduler tables. Multiple independent schedulers can share the same database if each scheduler specifies a different prefix string.

Multiple independent schedulers can share the same database if each instance specifies a different prefix string.

Important: Use a table prefix with all capital characters. If lowercase characters are used for the table prefix, they are automatically capitalized at run time.

Data type	String

Poll interval:

Specifies the interval, in seconds, that a scheduler polls the database. The default value is appropriate for most applications.

Each poll operation can be consuming. If the interval is extremely small and there are many scheduled tasks, polling can consume a large portion of system resources.

Data type	Integer
Units	Seconds
Default	30
Range	2000000

Work manager JNDI name:

Specifies the JNDI name of the work manager, which is used to manage the number of tasks that can run concurrently with the scheduler. The work manager also can limit the amount of Java Platform, Enterprise Edition (Java EE) context applied to the task.

The work manager is a server object that serves as a logical thread pool for the scheduler. Each repeating task that is created using this scheduler uses the **Number of alarm threads** specified in the work manager, which affects the number of tasks that can run concurrently. Use the work manager **Service Names** property to limit the amount of context information that is propagated to the task when it runs.

When a task runs, the task is run in the work manager associated with the scheduler instance. You can control the number of actively running tasks at a given time by configuring schedulers with a specific work manager. The number of tasks that can run concurrently is governed by the **Number of alarm threads** parameter on the work manager.

Use administration roles:

Specifies that when this option and administrative security are both enabled, the user administration roles are enforced when the scheduler JMX commands or APIs are used to create and modify tasks. If this option is not enabled, all the users can create and modify tasks.

Schedulers require several user roles to plan for, develop, administer and operate the scheduler service: administrator, developer and operator.

Data type	check box
Default	unchecked

Range	• Operator, Administrator - Calls any of the scheduler MBean or API methods and runs any of the scheduler administrative console functions. • Monitor, Configurator - Calls the scheduler MBean or API methods, but cannot create tasks or modify the state of any tasks. Only read-only methods and properties are accepted.

Configuring schedulers using Java Management Extensions

Schedulers can be created or configured using the Java Management Extensions (JMX) API using one of several scripting languages or Java.

About this task

To run with Java, the following JAR file needs to be present in the program class path: `com.ibm.ws.admin.client_6.1.0.jar`..

Complete these steps when using Java programs that utilize JMX.

Procedure

1. Look up the host and get an administration client handle.
2. Get a configuration service handle.
3. Update the `resource-pme.xml` file using the configuration service, as needed.
   a. Find the SchedulerProvider for a given scope.
   b. Create a SchedulerConfiguration and specify all required parameters identifying the SchedulerProvider as the parent object.
4. Reload the `resource-pme.xml` file to bind the newly created scheduler into the JNDI namespace. Perform this step if you want to use the newly created scheduler immediately, without restarting the application server.
   a. Locate the DataSourceConfigHelper MBean using the name.
   b. Invoke the reload() operation.

Results

A scheduler is now configured and ready to use for newly installed applications. If the scheduler JNDI name is not yet visible to your application, reinstalling the application or restarting the application server will allow the scheduler to be seen.

When schedulers are created for the first time, the poll daemon does not automatically start, and you must start it manually. When you restart the server, the poll daemon starts automatically. To start the poll daemon manually, refer to the scheduler daemons topic.

Note: Changes to existing scheduler configurations will not take affect until after the application server is restarted.

Example: Using scripting to create and configure schedulers:

Use the wsadmin scripting tool to invoke a `Jacl` script and create a SchedulerConfiguration resource.

The following `Jacl` example script can be invoked using the wsadmin scripting tool, which creates a SchedulerConfiguration resource using the DefaultWorkManager at the server scope.

```
# Example JACL Script to create a SchedulerConfiguration
# at the server scope

# Change the cell, node and server to match your environment
set cellName   MyCell
set nodeName   MyNode
set serverName server1

# We can just grab the first provider, since there is only one at the
# server scope level.

set schedProv [$AdminConfig getid /Cell:$cellName/Node:$nodeName/Server:$serverName/
  SchedulerProvider:SchedulerProvider]
if {$schedProv == ""} {
    puts "Unable to find SchedulerProvider for server: $serverName.  Aborting."
    exit
}
puts "Found a SchedulerProvider"

# Create a WorkManager for our scheduler at the server scope.
# We could use any of the other scopes as long as it is at the same
# or higher than the Scheduler's scope.

set wrkMgrProv [$AdminConfig getid /Cell:$cellName/Node:$nodeName/Server:$serverName/
  WorkManagerProvider:WorkManagerProvider/]
if {$wrkMgrProv == ""} {
    puts "Unable to find the WorkManagerProvider for server: $serverName.  Aborting."
    exit
}
puts "Found a WorkManagerProvider"

set wmName             "MyScheduler WorkManager"
set wmJNDIName         "wm/MySchedWorkManager"
set wmIsGrowable       false
set wmMaxThreads       1
set wmMinThreads       0
set wmNumAlarmThreads  10
set wmServiceNames     "com.ibm.ws.i18n;security;UserWorkArea;zos.wlm"
set wmThreadPriority   5

# Setup our DefaultWorkManager attributes
set createAttrs [subst { \
    {isGrowable $wmIsGrowable} \
    {jndiName $wmJNDIName} \
    {maxThreads $wmMaxThreads} \
    {minThreads $wmMinThreads} \
    {name "$wmName"} \
    {numAlarmThreads $wmNumAlarmThreads} \
    {serviceNames "$wmServiceNames"} \
    {threadPriority $wmThreadPriority} }]

puts "Creating a WorkManager"
$AdminConfig create WorkManagerInfo $wrkMgrProv $createAttrs
puts "WorkManager Created"

# Setup our SchedulerConfiguration attributes
set schedulerName          MyScheduler
set schedulerJNDIName      sched/MyScheduler
set datasourceJNDIName     jdbc/MySchedulerDatasource
set datasourceAlias        MySchedulerAlias
set pollInterval           30
set tablePrefix            MSCD
set useAdminRoles          true

set createAttrs [subst { \
  {name $schedulerName} \
  {datasourceJNDIName $datasourceJNDIName} \
```

```
  {datasourceAlias $datasourceAlias} \
  {jndiName $schedulerJNDIName} \
  {pollInterval $pollInterval} \
  {tablePrefix $tablePrefix} \
  {useAdminRoles true} \
  {workManagerInfoJNDIName $wmJNDIName}}]

puts "Creating a Scheduler"
$AdminConfig create SchedulerConfiguration $schedProv $createAttrs
puts "Scheduler created"

# Save the configuration
$AdminConfig save
```

Creating a scheduler resource reference

When you define schedulers in the server configuration, the object instance is bound into the global name space under the configured Java Naming Directory Interface (JNDI) name. You can use a resource reference to avoid manually coding this JNDI name into your application. Using a resource reference allows administrators to map applications to the appropriate schedulers.

About this task

You can alternatively create a scheduler resource reference by editing the XML directly. A Scheduler resource reference is a Java Platform, Enterprise Edition (Java EE) compliant resource that uses the class com.ibm.websphere.scheduler.Scheduler as the object type. For information regarding the XML file format, see the Java EE Specification.

Procedure

1. Start an assembly tool, such as Rational Application Developer.
2. Open the Java EE perspective.
3. Open your Enterprise JavaBeans (EJB) or Web module with the Deployment Descriptor Editor.
4. Click the **Reference** tab at the bottom of the window.
5. Click **Add**.
6. Select the **Resource reference** option.
7. Click **Next**.
8. Complete the Reference fields as shown in the following properties:
   Name The reference name, for example, *sched/MyScheduler*. According to this example, the name you choose has a local reference name of **java:comp/env/*sched/MyScheduler***.
   Type Select **com.ibm.websphere.scheduler.Scheduler**, and click **OK**.
   Authentication
   Select container.
   Description
   Any relevant description.
9. Click finish.
10. (**Optional**) Enter a global JNDI name of a configured scheduler in the JNDI name field in the **Bindings** section of the **Reference** window. You can specify or override this value when you install the application.
11. Save your changes to the deployment descriptor.

Results

A scheduler resource reference is now available to use within your application

Creating the database for schedulers

Each scheduler requires a database in which to store its persistent information. Schedulers use this database for storing tasks and then running them. The choice of database and location should be determined by the application developer and server administrator.

Before you begin

Scheduler performance is ultimately limited by database performance. If you need more tasks per second, you can run the scheduler daemons on larger systems, use clusters for the session beans used by the tasks or partition the tasks by using multiple schedulers. Eventually, however, the scheduler database becomes saturated, and a larger or better-tuned database system is needed. For detailed information on scheduler topologies refer to the "WebSphere Enterprise Scheduler planning and administration guide" technical paper.

Multiple schedulers can share a database when you specify unique table prefix values in each scheduler configuration. This sharing can lower the cost of administering scheduler databases. However, do not configure schedulers with non-unique table prefixes such that two separate servers share the same database table. A lease occurs between a specific database table and a scheduler running on a server. This lease allows one server at a time to own the lease to a specific database table. This process exists to ensure that one server runs schedule events, such as Enterprise JavaBeans (EJB) timers, in a cluster environment. If the server with this lease is unavailable, another server in the cluster obtains the lease.

About this task

Complete the following steps to create scheduler databases.

Procedure

1. Create a database. To create the database for a scheduler or to determine if an existing database is adequate for a scheduler, review the Create scheduler databases topic.
2. Create the scheduler tables. There are three methods for creating the tables for a scheduler:
   a. Create tables for schedulers using the administrative console. Use the administrative console to add, delete and verify database tables through your Web browser. This method is ideal for developers and simple scheduler topologies.
   b. Create tables for schedulers using JMX or scripting.

      Use JMX to add, delete and verify database tables programmatically with Java or scripting. This method is ideal for automating scheduler configurations for simple scheduler topologies.
   c. Create tables for schedulers using DDL files. Manually edit the DDL files using your favorite text editor, and verify that mapping between the table names and the scheduler resources and data sources is correct.

Creating scheduler databases

The scheduler uses the scheduler database for storing and running tasks. To create a scheduler database, your database system must be installed and available.

Before you begin

The performance of schedulers is ultimately limited by the performance of the database. If you need more tasks per second, you can run the scheduler daemons on larger systems or you can use clusters for the session beans used by the tasks. Eventually, however, the task database becomes saturated and you then need a larger or better-tuned database system.

Multiple applications can share a scheduler database. This sharing can lower the cost of administering scheduler databases.

About this task

The scheduler requires a database, a JDBC provider, and a data source.

Procedure

1. Create the database according to the description for your database system:
   - Creating Apache Derby databases for schedulers.
   - Creating a DB2 database for schedulers.
   - Creating a DB2 for iSeries database for schedulers.
   - Creating an Informix database for schedulers.
   - Creating a Microsoft SQL Server database for schedulers.
   - Creating an Oracle database for schedulers.
   - Creating a Sybase database for schedulers.
2. If the database is not on the same machine as your IBM WebSphere Application Server, verify that you can access the database from your application server machine.
3. Configure your JDBC provider and data source. For details, refer to the Creating and configuring a JDBC provider and data source topic. The JDBC driver can be either one-phase or two-phase commit depending on whether other transactions take place using other data sources, for example, while using the scheduler. The data source can represent multiple versions of the product.

Results

The database is created and ready for you to create scheduler tables.

Creating Apache Derby databases for schedulers:

This topic describes how to create Apache Derby databases for schedulers using data definition language (DDL) or structured query language (SQL) files.

About this task

To create Apache Derby databases for schedulers, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Verify that you have administrator rights for the database system.
3. Verify that the database supports Unicode (UTF-8). Otherwise, the database does not store all characters that can be handled in Java code, which results in code page conversion problems when a client uses an incompatible code page.
4. Use the ij utility supplied with the Apache Derby system to create the database. To use ij to create a database called scheddb which is located in `/opt`, for example, you would do the following:

   ```
   ij.sh
   ij>connect '/opt/scheddb;create=true';
   ij>quit;
   ```

   The embedded version of Apache Derby supports only one local connection. If the Application Server product is running and accessing a Apache Derby database, then any attempts to open a second connection to the database from the command line are rejected.

   Note: Add a semi-colon (;) at the end of each command. Otherwise, ij does not execute the command.

5. Exit the ij utility by issuing the quit command: `quit;`

Results

The Apache Derby database for the scheduler service exists.

Creating DB2 databases for schedulers:

This topic describes how to create DB2 databases for scheduler, using data definition language (DDL) or structured query language (SQL) files.

About this task

To create DB2 databases for scheduler, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a DB2 command-line window.
2. Make sure that you have administrator rights for the database system.
3. Verify that the database supports Unicode (UTF-8). Otherwise, it cannot store all the characters that can be handled in Java code, which might result in code page conversion problems, when a client uses an incompatible code page.

   To avoid deadlocks, be sure that the DB2 isolation level is set to "read stability". If necessary, enter the command

   ```
   db2set DB2_RR_TO_RS=YES
   ```

   then restart the DB2 instance to activate the change.
4. In the DB2 command line processor, enter this command to create the database with an example name, `scheddb`:

   ```
   db2 CREATE DATABASE scheddb USING CODESET UTF-8 TERRITORY en-us
   ```

   A DB2 database named `scheddb` has been created.

Results

The DB2 database for the scheduler exists.

Creating DB2 for iSeries databases for schedulers:

This topic describes how to create DB2 for iSeries databases for scheduler, using data definition language (DDL) or structured query language (SQL) files.

About this task

To create DB2 for iSeries databases for scheduler, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Run the following command to start an interactive SQL session:

   ```
   STRSQL
   ```
2. In interactive SQL, enter this command to create the collection with an example name, `scheddb`:

   ```
   CREATE COLLECTION scheddb
   ```
3. Exit the interactive SQL session.

4. Change the owner for the new collection to QEJBSVR by executing the following command:
   ```
   CHGOBJOWN OBJ(scheddb) OBJTYPE(*LIB) NEWOWN(QEJBSVR)
   ```

   where `scheddb` is the name of the collection you created in the previous step.

Results

The DB2 for iSeries database for the scheduler exists.

Creating Informix databases for schedulers:

This topic describes how to create Informix databases for schedulers, using data definition language (DDL) or structured query language (SQL) files.

About this task

To create Informix databases for schedulers, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Verify that you have administrator rights for the database system.
3. Verify that the database supports Unicode (UTF-8). Otherwise, the database does not store all characters that can be handled in the Java code, which results in code page conversion problems, when a client uses an incompatible code page.
4. If you want to create a new database named `scheddb`, for example, enter the command:
   ```
   dbaccess CREATE DATABASE scheddb with log
   ```

Results

The Informix database for scheduler exists.

Creating Microsoft SQL Server databases for schedulers:

This topic describes how to create Microsoft SQL Server databases for schedulers, using data definition language (DDL) or structured query language (SQL) files.

About this task

To create Microsoft SQL Server databases for schedulers, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Make sure that you are using a user ID that has administrator rights for the database system.
2. In the **Enterprise Manager**, expand a server group, then expand a server. A Microsoft SQL Server database named `scheddb` is created.
3. Right-click **Databases**, then click **New Database**.
4. Verify that the database supports Unicode (UTF-8). Otherwise, the database does not store all characters that can be handled in the Java code, which results in code page conversion problems, when a client uses an incompatible codepage.
5. Type the name `scheddb`.
6. Modify any default values, and save your changes. The Microsoft SQL Server database, scheddb, is created.

Results

The Microsoft SQL Server database for scheduler exists.

Creating Oracle databases for schedulers:

This topic describes how to create Oracle databases for schedulers, using data definition language (DDL) or structured query language (SQL) files.

About this task

To create Oracle databases for schedulers, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Make sure that you have administrator rights for the database system.
3. Verify that the database supports Unicode (UTF-8). Otherwise, the database does not store all characters that can be handled in the Java code, which results in code page conversion problems, when a client uses an incompatible code page.
4. Use the Database Configuration Assistant to create the database, `scheddb`, for example. Verify that you select the **JServer** option for the database. Use a Unicode code page when creating the database. The text data you pass to the APIs must be compatible with the selected code page.

Results

The Oracle database for scheduler exists.

Creating Sybase databases for schedulers:

This topic describes how to create Sybase databases for schedulers, using data definition language (DDL) or structured query language (SQL) files.

About this task

This topic describes how to create Sybase databases for schedulers, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Make sure that you have administrator rights for the database system.
3. Make sure that you have the DTM option for Sybase ASE installed.
4. Verify that the database supports Unicode (UTF-8). Otherwise, the database does not store all characters that can be handled in the Java code, which results in code page conversion problems, when a client uses an incompatible code page.
5. Use the Sybase isql utility to create the database, `scheddb`, for example. See your Sybase product documentation for details.

Results

The Sybase database for the scheduler exists.

Scheduler table management functions

The administration console and the WASSchedulerConfiguration MBeans provide simplified methods for creating scheduler tables and schema, verifying that the scheduler tables and schema are setup properly and are accessible and removing scheduler tables and schema.

Note: There are limitations when running the table management functions relating to data source access. See the Verifying a connection topic for details on these limitations. If a connection cannot be verified successfully, the scheduler table management functions will fail.

Verify tables

Validates that scheduler data sources, table prefixes, security authentication information and tables are configured correctly. You can use this verification method in production and development environments without altering database properties.

Create tables

Creates the necessary tables and indices required for schedulers to operate. This method of creating scheduler tables is designed for simple topologies and development environments. Use the supplied scheduler data definition language files for advanced or production environments and for databases that do not support this feature. For details, see the topic Creating scheduler tables using the administrative console.

Drop tables

Specifies the removal of tables and indices required for schedulers to operate. This method of removing scheduler tables and indices is recommended for development environments. When you drop tables, the action removes all previously scheduled tasks, and the scheduler no longer operates successfully, until the tables are recreated.

Scheduler table definition

Schedulers require database tables and indices with a table prefix. This page provides reference information about the tables.

Each scheduler requires several database tables and indices to operate. Each table name and index described in this topic requires a table prefix. For example, if the scheduler is configured with a table prefix value, **SCHED_**, the table with the name, **TASK**, would be named **SCHED_TASK**. See Scheduler settings for details on the table prefix.

To create the tables, see Creating the database for schedulers. To see the exact schema definition such as field sizes and types, see Creating scheduler tables using DDL files. This section references the location where the DDL or SQL statements are stored. These statements create the table schema.

Note: The information in this topic is provided for problem determination. Do not alter the scheduler table names, field names or index names. The data content format might change without notice. Be aware of this factor when accessing the tables directly. Modifying data in the tables without using the Scheduler API might cause failures.

TASK

The TASK table contains the tasks that have been scheduled, but not yet purged. The primary key for this table is the TASKID which equates to the getTaskID() method on the com.ibm.websphere.scheduler.TaskStatus interface.

Since there is one row in this table for each task, it is important that the database and table support row-locking. Using page, or table locks, inhibits the scheduler from running tasks concurrently.

Table 71. TASK table.. Displays scheduled tasks

Field name	Purpose and notes
TASKID	Contains all of the tasks that have been scheduled, but not yet purged. The primary key for this table is TASKID which equates to the getTaskID() method on the com.ibm.websphere.scheduler.TaskStatus interface. Since there is one row in this table for each task, it is important that the database and table support row-locking. Using page, or table locks, will inhibit the scheduler from running tasks concurrently.
VERSION	Internal version ID of this row format.
ROW_VERSION	The version of this row. Used for optimistic locking.
TASKTYPE	The type of task: **1**=BeanTaskInfo, **2**=MessageTaskInfo
TASKSUSPENDED	This value indicates if the task is suspended or if it is running. The task is suspended if the value BITWISE AND 1 equals **1**. The task is running if the value BITWISE AND 2 equals **2**.
CANCELLED	The value, **1**, if the task is cancelled.
NEXTFIRETIME	The date in milliseconds using java.util.Date.getTime() when the task is scheduled to run next.
STARTBYINTERVAL	The start-by-interval of the task.
STARTBYTIME	Reserved.
VALIDFROMTIME	The task start time.
VALIDTOTIME	Reserved.
REPEATINTERVAL	The task repeat interval.
MAXREPEATS	The number of times to run the task.
REPEATSLEFT	The number of times the task has yet to run.
TASKINFO	Internal binary data.
NAME	The task name.
AUTOPURGE	The value, **1**, if the task is to automatically purge upon completion.
FAILUREACTION	Reserved.
MAXATTEMPTS	Reserved.
QOS	Reserved.
PARTITIONID	Reserved.
OWNERTOKEN	The task owner.
CREATETIME	The time in milliseconds using java.util.Date.getTime() when the task was created.

The TASK table also has the following indices that are required to allow the scheduler to run and access tasks concurrently:

- TASK_IDX1 – Used to access individual tasks using the Scheduler API.
- TASK_IDX2 – Used by the poll daemon to load expiring tasks.

TREG

The TREG table is used to store scheduler information that is shared between redundant schedulers. This table is not highly used.

Table 72. TREG table.. Displays scheduler information between redundant schedulers

Field name	Purpose and notes
REGKEY	The registry key. This is the primary key of the table.

Table 72. TREG table. (continued). Displays scheduler information between redundant schedulers

REGVALUE	The registry value.

LMGR

The LMGR table is used to track the leases that redundant schedulers use. This table is not highly used.

Table 73. LMGR table.. Displays redundant scheduler leases

Field name	Purpose and notes
LEASENAME	The name of the lease. This is the scheduler JNDI name and is the primary key.
LEASEOWNER	The owner of the lease. The format is `Cell/Node/Server`.
LEASE_EXPIRE_TIME	The time in milliseconds using java.util.Date.getTime() when the lease for the scheduler expires.
DISABLED	Reserved.

LMPR

The LMPR table is used to store arbitrary properties for the lease. This table is not highly utilized.

Table 74. LMPR table.. Displays arbitrary lease properties

Field name	Purpose and notes
LEASENAME	The name of the lease. See the LMGR table.
NAME	The name of the property.
VALUE	The value of the property.

The LMPR table also has the following index:

- LMPR_IDX1 – Used to retrieve properties for a given lease.

Creating scheduler tables using the administrative console

To create scheduler tables using the administrative console, the scheduler requires a database, a Java DataBase Connectivity (JDBC) provider and a data source.

Before you begin

Note: Limitations for Oracle XA databases

> Oracle XA prohibits required schema operations in a global transaction environment. Local transactions are not supported. If you have schedulers that use an Oracle XA data source, either temporarily change the scheduler configuration to use a non-XA Oracle data source, or create the tables manually using the supplied DDL files. If you use the administrative console to create or drop scheduler tables for a scheduler configured to use an Oracle XA data source, then you receive a SchedulerDataStoreException error message, and the operation fails.

Note: Limitations for DB2 z/OS databases

Creating and dropping tables using the administrative console is not supported for DB2 z/OS databases. A database administrator is typically involved with defining and managing databases on DB2 z/OS systems. The administration interface is targeted for the non-database administrator or developer who does not want to know the specifics of setting up the scheduler database. The scheduler has DDL files available for the database administrator to create the required tables.

Procedure

1. Verify that the database to be used for this scheduler is available and accessible by the application server. Review the Creating scheduler databases and tables topic for instructions on creating a database. The remaining steps describe how to create scheduler tables in an existing database.
2. Start the administrative console.
3. Create a JDBC data source that refers to the scheduler database.
4. Test the data source connection.
5. Create a scheduler. This configuration object contains the desired table prefix and the JNDI name of the JDBC data source. Verify that you save the new Scheduler to the configuration repository before you proceed to the next step.
6. Click **Resources** > **Schedulers** to view all defined schedulers.
7. Select one or more schedulers.
8. Click **Create Tables** to create the tables for the selected schedulers in their associated database. The tables and indices you created reflect the table prefixes and data sources specified in each scheduler configuration.
9. Restart the server or start the poll daemon to run scheduler tasks.

Results

Scheduler tables and schema are created.

Creating scheduler tables using scripting and Java Management Extensions

Creating scheduler tables using scripting and Java Management Extensions requires a database, a Java DataBase Connectivity (JDBC) provider, and a data source.

Before you begin

Attention: Limitations for Oracle XA databases

Oracle XA prohibits required schema operations in a global transaction environment. Local transactions are not supported. If you have schedulers that use an Oracle XA data source, either temporarily change the scheduler configuration to use a non-XA Oracle data source, or create the tables manually using the supplied DDL files. If you use the administrative console to create or drop scheduler tables for a scheduler configured to use an Oracle XA data source, then you receive a SchedulerDataStoreException error message, and the operation fails.

Attention: Limitations for DB2 z/OS databases

Creating and dropping tables using the administrative console is not supported for DB2 z/OS databases. A database administrator is typically involved with defining and managing databases on DB2 z/OS systems. The administration interface is targeted for the non-database administrator or developer who does not want to know the specifics of setting up the scheduler database. The scheduler has DDL files available for the database administrator to create the required tables.

Procedure

1. Verify that the database to be used for this Scheduler is available and accessible by the application server. Review the Creating scheduler databases and tables topic for instructions on creating a database. The remainder of these steps describe how to create scheduler tables in an existing database.
2. Launch the wsadmin tool and connect to an application server. This process requires an active server to be available and fails, if you are disconnected from the server.
3. Create a JDBC data source that refers to the scheduler database.
4. Test the data source connection.

5. Create a scheduler. This configuration object contains the desired table prefix and the JNDI name of the JDBC data source. Verify that you save the new Scheduler to the configuration before you proceed to the next step.
6. Run the **createTables** MBean operation.
   a. Look up the SchedulerConfiguration object or use the object you created in a previous step.
   b. Locate the **WASSchedulerConfiguration** MBean.
   c. Run one of the **createTables** MBean operation on the **WASSchedulerConfiguration** object to create the tables for the specified **SchedulerConfiguration** object in its associated database. The tables and indices that you created reflect the table prefix and data source specified in the scheduler configuration.
7. Restart the server or start the poll daemon to run scheduler tasks.

Results

Scheduler tables and schema are created.

Example: Using scripting to verify scheduler tables:

Use the wsadmin scripting tool to invoke a `Jacl` script and verify tables and indices for a scheduler.

The following `Jacl` example script can be invoked using the wsadmin scripting tool, which verifies that the tables and indices are created correctly for a scheduler. See the “Configuring Schedulers” topic for details on how a scheduler is created.

```
# Example JACL Script to verify the scheduler tables

# The name of the scheduler to verify
set schedName "My Scheduler"

puts ""
puts "Looking-up Scheduler Configuration Helper MBean"
puts ""
set schedHelper [$AdminControl queryNames WebSphere:*,type=WASSchedulerCfgHelper]

#Access the configuration object.
set myScheduler [$AdminConfig getid /SchedulerConfiguration:$schedName/]

if {$myScheduler == ""} {
    puts ""
    puts "Error: Scheduler $schedName could not be found."
    puts ""
    exit
}

# Invoke the verifyTables method on the helper MBean.

puts ""
puts "Verifying tables for:"
puts "$myScheduler"
puts ""

if { [catch {$AdminControl invoke $schedHelper verifyTables $myScheduler} errorInfo] } {
    puts ""
    puts "Error verifying tables: $errorInfo"
    puts ""
} else {
    puts ""
    puts "Tables verified successfully."
    puts ""
}
```

Example: Using scripting to create scheduler tables:

Use the wsadmin scripting tool to invoke a Jacl script and create scheduler tables.

The following Jacl example script can be invoked using the wsadmin scripting tool, which creates the scheduler tables for a configured scheduler. See the Configuring Schedulers topic for details on how to create a scheduler.

```
# Example JACL Script to create the scheduler tables

# The name of the scheduler to create tables for
set schedName "My Scheduler"

puts ""
puts "Looking-up Scheduler Configuration Helper MBean"
puts ""
set schedHelper [$AdminControl queryNames WebSphere:*,type=WASSchedulerCfgHelper]

#Access the configuration object.
set myScheduler [$AdminConfig getid /SchedulerConfiguration:$schedName/]

if {$myScheduler == ""} {
    puts ""
    puts "Error: Scheduler with name: $schedName could not be found."
    puts ""
    exit
}

# Invoke the createTables method on the helper MBean.

puts ""
puts "Creating tables for:"
puts "$myScheduler"
puts ""

if {[catch {
    set result [$AdminControl invoke $schedHelper createTables $myScheduler]
    if {$result} {
        puts ""
        puts "Successfully created the tables."
        puts ""
    } else {
        puts ""
        puts "The tables were already created."
        puts ""
    }
  } errorInfo ] } {
    puts ""
    puts $errorInfo
    puts ""
}
```

Example: Using scripting to drop scheduler tables:

Use the wsadmin scripting tool to invoke a Jacl script and remove scheduler tables.

The following Jacl example script can be invoked using the wsadmin scripting tool, which removes the scheduler tables for a configured scheduler. See the “Configuring Schedulers” topic for details on how a scheduler is created

```
# Example JACL Script to drop the scheduler tables

# The name of the scheduler to drop the tables for
set schedName "My Scheduler"

puts ""
puts "Looking-up Scheduler Configuration Helper MBean"
puts ""
```

```
set schedHelper [$AdminControl queryNames WebSphere:*,type=WASSchedulerCfgHelper]

#Access the configuration object.
set myScheduler [$AdminConfig getid /SchedulerConfiguration:$schedName/]

if {$myScheduler == ""} {
    puts ""
    puts "Error: Scheduler with name: $schedName could not be found."
    puts ""
    exit
}

# Invoke the dropTables method on the helper MBean.

puts ""
puts "Dropping tables for:"
puts "$myScheduler"
puts ""

if {[catch {
    set result [$AdminControl invoke $schedHelper dropTables $myScheduler]
    if {$result} {
        puts ""
        puts "Successfully dropped the tables."
        puts ""
    } else {
        puts ""
        puts "The tables were already dropped."
        puts ""
    }
  } errorInfo ] } {
    puts ""
    puts $errorInfo
    puts ""
}
```

Creating scheduler tables using DDL files

This topic provides the steps for creating scheduler tables using DDL files.

Before you begin

Your database system must be installed and available.

The scheduler requires a database, a Java™ Database Connectivity (JDBC) provider, and a data source.

About this task

Complete the following steps to create scheduler tables using DDL files.

Procedure

1. Verify that your database is created. Refer to the Creating scheduler databases topic.
2. Create the database tables according to the instructions for your database system:
   - Creating Apache Derby tables for schedulers
   - Creating DB2 tables for schedulers.
   - Creating DB2 tables for z/OS for schedulers.
   - Creating DB2 for iSeries tables for schedulers.
   - Creating Informix tables for schedulers.
   - Creating Microsoft SQL Server tables for schedulers.
   - Creating Oracle tables for schedulers.

- Creating Sybase tables for schedulers.

The following scripts are deprecated, but will still work:

Table 75. Deprecated scripts.. Deprecated DDL scripts

Deprecated script	New script
createSchemaDB2.ddl	createSchemaMod1DB2.ddl
createSchemaDB2iSeries.ddl	createSchemaMod1DB2iSeries.ddl
createSchemaDerby.ddl	createSchemaMod1Derby.ddl
createSchemaMSSQL.sql	createSchemaMod1MSSQL.sql
createSchemaInformix.sql	createSchemaMod1Informix.sql
createSchemaOracle.ddl	createSchemaMod1Oracle.ddl
createSchemaSybase12.ddl	createSchemaMod1Sybase12.ddl
dropSchemaMSSQL.sql	dropSchemaMod1MSSQL.sql
dropSchemaDB2.ddl	dropSchemaMod1DB2.ddl
dropSchemaDB2iSeries.ddl	dropSchemaMod1DB2iSeries.ddl
dropSchemaDerby.ddl	dropSchemaMod1Derby.ddl
dropSchemaInformix.sql	dropSchemaMod1Informix.sql
dropSchemaOracle.ddl	dropSchemaMod1Oracle.ddl
dropSchemaSybase12.ddl	dropSchemaMod1Sybase12.ddl

Creating Apache Derby tables for schedulers:

This topic describes how to create tables for schedulers on Apache Derby databases, using data definition language (DDL) or structured query language (SQL) files.

Before you begin

This task requires you to configure a database and make it available. Refer to the Creating Cloudscape databases for schedulers topic, for more information.

About this task

To create tables for schedulers on Apache Derby databases, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Create the schema.
   a. Using a text editor, edit the script, *%app_server_root%*\Scheduler\createSchemaMod1Derby.ddl, according to the instructions at the top of the file.

      Attention: When setting the table prefix, capitalize all characters.
   b. Enter one of the following commands.

      Attention: Apache Derby provides both an embedded and network server version. This example is for the embedded version of Apache Derby. See the Apache Derby product documentation for more details on running DDL scripts.
      On Windows systems (using the example name, scheddb):

      ```
      %app_server_root%\derby\bin\embedded\ij.bat %app_server_root%\Scheduler\createSchemaMod1Derby.ddl
      ```

On UNIX systems (using the example name, scheddb):

```
%app_server_root%/derby/bin/embedded/ij.sh %app_server_root%/Scheduler/createSchemaMod1Derby.ddl
```

Results

The Apache Derby tables and schema for the scheduler exist.

Creating DB2 tables for schedulers:

This topic describes how to create tables for scheduler on DB2 databases, using data definition language (DDL) or structured query language (SQL) files.

Before you begin

This task requires you to configure a database and make it available. Refer to the Creating DB2 databases for schedulers topic for more information.

About this task

To create tables for scheduler on DB2 databases, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a DB2 command-line window.
2. Verify that you have administrator rights for the database system.
3. Create the table space and schema.
   a. Analyze the results of your experiences during development and system testing. The size of your database depends on many factors. If possible, distribute table space containers across different logical disks, and implement an appropriate security policy. Consider the performance implications of your choices for buffer pools and log file settings.
   b. Using a text editor, edit the following scripts according to the instruction at the top of each file.

      Note: When setting the table prefix, capitalize all characters.

      `%WAS_HOME%\Scheduler\createTablespaceDB2.ddl, %WAS_HOME%\Scheduler\createSchemaMod1DB2.ddl, %WAS_HOME%\Scheduler\dropSchemaMod1DB2.ddl, and %WAS_HOME%\Scheduler\dropTablespaceDB2.ddl.`
   c. Verify that you are attached to the correct instance. Check the environment variable DB2INSTANCE.
   d. To connect to the database, scheddb, for example, and enter the command:

      ```
      db2 connect to scheddb
      ```
   e. Create the table space. Enter the following command:

      ```
      db2 -tf createTablespaceDB2.ddl
      ```

      Verify that the script output contains no errors. If there were any errors, you can drop the table space using the following script:

      ```
      dropTablespaceDB2.ddl
      ```
   f. To create the schema (tables and indices), in the DB2 command line processor, enter the command `db2 -tf createSchemaMod1DB2.ddl`. Verify that the script output contains no errors. If there were any errors, you can use the following file to drop the schema:

      ```
      dropSchemaMod1DB2.ddl
      ```
   g. Verify that the DB2_RR_TO_RS DB2 flag is set to **YES** to avoid deadlocks. Restart the DB2 instance to activate the change, if needed.

Results

The DB2 tables and schema for the scheduler exist.

Creating DB2 for iSeries tables for schedulers:

This topic describes how to create tables for scheduler on DB2 for iSeries databases, using data definition language (DDL) or structured query language (SQL) files.

Before you begin

This task requires you to configure a database and make it available. See the " Creating DB2 for iSeries databases for schedulers" topic for more information.

About this task

To create tables for scheduler on DB2 for iSeries databases, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Start a QShell session by running the following command from a CL command line:
   ```
   STRQSH
   ```
2. Create a new IFS directory under your profile directory:
   ```
   mkdir profile_root/scheduler
   ```
   Where *profile_root* is the fully qualified path of the directory that contains your profile.
3. Copy the following scripts to this new directory:
   ```
   %WAS_HOME%/scheduler/createDB2iSeriesSchema.ddl
   %WAS_HOME%/scheduler/dropSchemaDB2iSeries.ddl
   ```
4. Using a text editor, edit these scripts in the new directory. Replace all occurrences of @TABLE_PREFIX@ with <*collection_name*>, <*table_prefix*> where <*collection_name*> is the name of the collection that will be used to store the database files and is the collection that was created in the " Creating DB2 for iSeries databases for schedulers" topic. The <*table_prefix*> precedes each table name.
5. Use the iSeries Navigator to create the database files on your iSeries server.
   a. Start iSeries Navigator.
   b. Expand the iSeries icon to locate the system where you want to create the database files.
   c. Expand **Database**, and right-click the system database.
   d. Select **Run SQL Scripts**.
   e. Select **File** > **Open**.
   f. Navigate to the `createDB2iSeriesSchema.ddl` file in your profile directory.
   g. Select **Run** > **All**.
   h. Select **View** > **Job Log** to verify that the tables were created.
6. Execute the following command to change the owner for the new database files to QEJBSVR:
   ```
   CHGOBJOWN OBJ(*ALL) OBJTYPE(*ALL) NEWOWN(QEJBSVR)
   ```

Results

The DB2 for iSeries tables and schema for the scheduler exist.

Creating Informix tables for schedulers:

This topic describes how to create tables for schedulers on Informix databases, using data definition language (DDL) or structured query language (SQL) files.

Before you begin

This task requires that you configure a database and make it available. Refer to the Creating Informix databases for schedulers topic for more information.

About this task

To create tables for schedulers on Informix databases, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Verify that you have administrator rights for the database system.
3. Create the schema.
   a. Using a text editor, edit the script `%WAS_HOME%\Scheduler\createSchemaMod1Informix.sql` according to the instruction at the top of the file.

      Note: When setting the table prefix, capitalize all characters.
   b. Enter the following command, using the database, `scheddb`, for example:

      ```
      dbaccess scheddb createSchemaMod1Informix.sql
      ```

Results

The Informix tables and schema for the scheduler exist.

Creating Microsoft SQL Server tables for schedulers:

This topic describes how to create tables for schedulers on Microsoft SQL Server databases, using data definition language (DDL) or structured query language (SQL) files.

Before you begin

This task requires you to configure a database and make it available. Refer to the Creating Microsoft SQL Server databases for schedulers topic for more information.

About this task

To create tables for schedulers on Microsoft SQL Server databases, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Change to the directory where the configuration scripts for scheduler are located. This is the Scheduler subdirectory of the IBM WebSphere Application Server installation directory.
3. Use a text editor to edit the schema creation script, `createSchemaMod1MSSQL.sql`, according to the instructions at the beginning of the file.

   Tip: When setting the table prefix, capitalize all characters.
4. Create the schema:

a. Verify that you have administrator rights for the database system. The user ID you use to create the schema must be the one that you configure WebSphere Application Server to use when accessing the database.
b. Run the following script to create the schema (tables and views) :

```
sqlcmd -S <servername> -U<userid> -P<password> -d<databaseName> -i<script name>
```

Results

The Microsoft SQL Server tables and schema for scheduler exist.

Creating Oracle tables for schedulers:

This topic describes how to create tables for schedulers on Oracle databases, using data definition language (DDL) or structured query language (SQL) files.

Before you begin

This task requires you to configure a database and make it available. Refer to the Creating Oracle databases for schedulers topic for more information.

About this task

To create tables for schedulers on Oracle databases, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Make sure that you have administrator rights for the database system.
3. Create the table space and schema.
   a. Using a text editor, edit the following scripts according to the instructions at the top of the files.

      Tip: When setting the table prefix, capitalize all characters.

      %app_server_root%\Scheduler\createTablespaceOracle.ddl and *%app_server_root%*\createSchemaMod1Oracle.ddl
   b. Set the environment variable ORACLE_SID, if you do not want the schema to be created in the default instance.
   c. Run the script, createTablespaceOracle.ddl, to create the table space.

      For test purposes, use the same location for all table spaces and pass the path as a command line argument to the script.
   d. Run the script, createSchemaMod1Oracle.ddl, to create the schema.

Results

The Oracle tables and schema for scheduler exist.

Creating Sybase tables for schedulers:

This topic describes how to create tables for schedulers on Sybase databases, using data definition language (DDL) or structured query language (SQL) files.

Before you begin

This task requires you to configure a database and make it available. Refer to the Creating Sybase databases for schedulers topic for more information.

About this task

To create tables for schedulers on Sybase databases, using data definition language (DDL) or structured query language (SQL) files, use these steps.

Procedure

1. Open a command-line window.
2. Make sure that you have administrator rights for the database system.
3. Make sure that you have the Distributed Transaction Management (DTM) option for Sybase ASE installed.
   a. Set enable DTM to **1** in the Sybase server configuration.
   b. Set enable xact coordination to **1** in the Sybase server configuration.
   c. Add the **dtm_tm_role** role to the Sybase administration user ID. For example, enter the user ID `sa`.
   d. Restart the Sybase server.
4. Use the Sybase isql utility to create a database. For example, enter the database name `scheddb`. See your Sybase product documentation for details.
5. Create the schema:
   a. Using a text editor, edit the following script according to the instructions located at the top of the file.

      Tip: When setting the table prefix, capitalize all characters.

      app_server_root`\Scheduler\createSchemaMod1Sybase12.ddl`
   b. Enter the following command:

      ```
      isql -S <servername> -U <userid> -P <password> -D scheddb -i createSchemaMod1Sybase12.ddl
      ```

Results

The Sybase tables and schema for scheduler exist.

Chapter 23. Administering application security

Setting up, enabling and migrating security

You must address several issues prior to authenticating users, authorizing access to resources, securing applications, and securing communications. These security issues include migration, interoperability, and installation.

About this task

After installing WebSphere Application Server, you can determine the proper level of security that is needed for your environment. By default, administrative security is enabled and provides the authentication of users using the WebSphere administration functions, the use of Secure Sockets Layer (SSL), and the choice of user account repository.

You can also use the following permissions to enhance security:

- Use the getSSLConfig permission to give your application code the ability to call several of the JSSEHelper methods. For more information about these methods, see the description of the com.ibm.websphere.ssl.JSSEHelper API in the Programming interfaces section of the Information Center.
- Use the AdminPermission permission to give your application code the ability to call WebSphere Application Server administrative APIs. See the topic Setting Java 2 security permissions for an example of how to set this permission.
- Use the accessRuntimeClasses permission to give your application code the ability to load classes that are included with the product. If you are operating in an environment that normally restricts access to these classes, this permission enables your application code to bypass this restriction during class loading. See the topic Global security settings for a description of how to set this permission.

The following information is covered in this section:

Procedure

Enable security for all your application servers or for specific application servers in your realm.
For more information, see “Enabling security” on page 1172.

What to do next

After installing WebSphere Application Server and securing your environment, you must authenticate users. For more information, see “Authenticating users” on page 1254.

Migrating, coexisting, and interoperating – Security considerations

Use this topic to migrate the security configuration of previous WebSphere Application Server releases and its applications to the new installation of WebSphere Application Server.

Before you begin

This information addresses the need to migrate your security configurations from a previous release of IBM WebSphere Application Server to WebSphere Application Server 8.0. Complete the following steps to migrate your security configurations:

- If security is enabled in the previous release, obtain the administrative server ID and password of the previous release. This information is needed in order to run certain migration jobs.
- You can optionally disable security in the previous release before migrating the installation. No logon is required during the installation.

Note: In WebSphere Application Server Version 8.0, be aware of the following additional migration requirements for security:

- When migrating from WebSphere Application Server Version 7.x to Version 8.0, if you have a business need to preserve security audit logs from the older release you must first archive the security audit log files in Version 7.x. WebSphere Application Server does not support the migration of security audit log files from the older release to Version 8.0.
- If your WebSphere Application Server Version 7.x environment is enabled for Kerberos, and you are migrating to version 8.0 on a different machine, the keytab and configuration files for Kerberos must be at the same location on the Version 8.0 machine as on the Version 7.x machine or the configuration will not work.

Procedure

Follow the steps in "Migrating product configurations".

Results

The security configuration of previous WebSphere Application Server releases and its applications are migrated to the new installation of WebSphere Application Server Version 8.0.

What to do next

You must migrate any custom class files that are not migrated.

If the previous version instance is configured to enable secure connections using digital certificates that are signed by the Digital Certificate Manager (DCM) local certificate authority, those certificates must be renewed. For example, they must be renewed for the previous version instance, the WebSphere Application Server Version 8.0 profile, and all of the Secure Socket Layer-enabled clients and servers that connect to WebSphere Application Server. For more information, see SSL handshake failure using digital certificates signed by a Digital Certificate Manager (DCM) local certificate authority.

IBM i *SYSTEM certificate stores for applications are deprecated in WebSphere Application Server Version 5. In WebSphere Application Server Version 8.0, you must migrate your applications to use Java keystores.

Interoperating with previous product versions

IBM WebSphere Application Server inter-operates with the previous product versions. Use this topic to configure this behavior.

Before you begin

The current release of the Application Server distinguishes the identities of the user who acts as an administrator, managing the Application Server environment, from the identity of the user that is used for authenticating between servers. In prior releases, the end user had to specify a server user ID and password as the user identity for authenticating between servers. In the current release of the Application Server, the server user ID is generated automatically and internally; however, the end user can specify that the server user ID and password not be automatically generated. This option is especially important in the case of a mixed-release cell, where the server user ID and password are specified in a down-level version of the Application Server. In such a scenario, the end user should opt out of automatically generating the server user ID and instead use the server user ID and password that is specified in the down-level version of the Application Server, in order to ensure backwards compatibility.

Interoperability is achieved only when the Lightweight Third Party Authentication (LTPA) authentication mechanism and a distributed user registry is used such as Lightweight Directory Access Protocol (LDAP) or a distributed Custom user registry. LocalOS on most platforms is not considered a distributed user registry (except on z/OS within the z/OS environment).

Procedure

1. Configure WebSphere Application Server Version 8.0 with the same distributed user registry (that is, LDAP or Custom) that is configured with the previous version. Make sure that the same LDAP user registry is shared by all of the product versions.
   a. In the administrative console, select **Security > Global security**.
   b. Choose an available Realm definition and click **Configure**.
   c. Enter a **Primary administrative user name**. This identity is the user with administrative privileges that is defined in your local operating system. If you are not using the local OS ad the user registry, select the **Server identity that is stored in the user repository**, enter the Server user ID, and the associated password. The user name is used to log on to the administrative console when administrative security is enabled. WebSphere Application Server Version 6.1 requires an administrative user that is distinct from the server user identity so that administrative actions can be audited.

      Attention: In WebSphere Application Server, Version 6.0.x, a single user identity is required for both administrative access and internal process communication. When migrating to Version 8.0, this identity is used as the server user identity. You need to specify another user for the administrative user identity.
   d. When interoperating with Version 6.0.x or previous versions, you must select the Server identity that is stored in the user repository. Enter the **Server user id** and the associated **Password.**
2. Configure the LTPA authentication mechanism. Automatic generation of the LTPA keys should be disabled. If not, keys used by a previous release are lost. Export the current LTPA keys from WebSphere Application Server Version 8.0 and import them into the previous release.
   a. In the administrative console select **Security > Global security**.
   b. From Authentication mechanisms and expiration, click **LTPA**.
   c. Click the **Key set groups** link , then click the key set group that displays in the Key set groups panel.
   d. Clear the **Automatically generate keys** check box.
   e. Click **OK**, then click **Authentication mechanisms and expiration** in the path at the top of the Key set groups panel.
   f. Scroll down to the Cross-cell single sign-on section, and enter a password to use for encrypting the LTPA keys when adding them to the file.
   g. Enter the password again to confirm the password.
   h. Enter the **Fully qualified key file name** that contains the exported keys.
   i. Click **Export keys**.
   j. Follow the instructions provided in the previous release to import the exported LTPA keys into that configuration.
3. If you are using the default SSL configuration, extract all of the signer certificates from the WebSphere Application Server Version 8.0 common trust store. Otherwise, extract signers where necessary to import them into the previous release.
   a. In the administrative console, click **Security > SSL certificate and key management**.
   b. Click **Key stores and certificates**.
   c. Click **NodeDefaultTrustStore**.
   d. Click **Signer certificates**.
   e. Select one signer and click **Extract**.
   f. Enter a unique path and filename for the signer. For example, `/tmp/signer1.arm`.
   g. Click **OK**. Repeat for all of the signers in the trust store.
   h. Check other trust stores for other signers that might need to be shared with the other server. Repeat steps e through h to extract the other signers.

You can also import a signer certificate, which is also called a certificate authority (CA) certificate, from a truststore on a non-z/OS platform server to a z/OS keyring. the z/OS keyring contains the signer certificates that originated on the non-z/OS platform server. For more information, see Importing a signer certificate from a truststore to a z/OS keyring.

4. Add the exported signers to `DummyServerTrustFile.jks` and `DummyClientTrustFile.jks` in the `/etc` directory of the back-level product version. If the previous release is not using the dummy certificate, the signer certificate(s) from the previous release must be extracted and added into the WebSphere Application Server Version 8.0 release to enable SSL connectivity in both directions.
   a. Open the key management utility, iKeyman, for that product version.
   b. Start `ikeyman.bat` or `ikeyman.sh` from the `${USER_INSTALL_ROOT}/bin` directory.
   c. Select **Key Database File > Open**.
   d. Open `${USER_INSTALL_ROOT}/etc/DummyServerTrustFile.jks`.
   e. Enter WebAS for the password.
   f. Select **Add** and enter one of the files extracted in step 2. Continue until you have added all of the signers.
   g. Repeat steps c through f for the `DummyClientTrustFile.jks` file.
5. Verify that the application uses the correct Java Naming and Directory Interface (JNDI) name and naming bootstrap port for performing a naming lookup.
6. Stop and restart all of the servers.

Migrating trust association interceptors

Use this topic to manually migrate trust associations.

Before you begin

Note: Data sources are not supported for use within a Trust Association Interceptor (TAI). Data sources are intended for use within J2EE applications and designed to operate within the EJB and web containers. Trust Association Interceptors do not run within a container, and while data sources may function in the TAI environment, they are untested and not guaranteed to function properly.

The following topics are addressed in this document:
- Changes to the product-provided trust association interceptors
- Migrating product-provided trust association interceptors
- Changes to the custom trust association interceptors
- Migrating custom trust association interceptors

Changes to the product-provided trust association interceptors

For the product-provided implementation for the WebSEAL server, a new optional com.ibm.websphere.security.webseal.ignoreProxy property is added. If this property is set to `true` or `yes`, the implementation does not check for the proxy host names and the proxy ports to match any of the host names and ports that are listed in the com.ibm.websphere.security.webseal.hostnames and the com.ibm.websphere.security.webseal.ports property respectively. For example, if the VIA header contains the following information:

```
HTTP/1.1 Fred (Proxy), 1.1 Sam (Apache/1.1),
HTTP/1.1 webseal1:7002, 1.1 webseal2:7001
```

and the com.ibm.websphere.security.webseal.ignoreProxy property is set to `true` or `yes`, the host name `Fred`, is not used when matching the host names. By default, this property is not set, which implies that any proxy host names and ports that are expected in the VIA header are listed in the host names and the ports properties to satisfy the isTargetInterceptor method.

The previous VIA header information was split onto two lines for illustrative purposes only.

For more information about the com.ibm.websphere.security.webseal.ignoreProxy property, see the article in the information center on configuring single signon using trust association interceptor ++.

Migrating product-provided trust association interceptors

The properties that are located in the `webseal.properties` and `trustedserver.properties` files are not migrated from previous versions of WebSphere Application Server. You must migrate the appropriate properties to WebSphere Application Server Version 6.0.*x* using the trust association panels in the administrative console. For more information, see Configuring trust association interceptors.

Changes to the custom trust association interceptors

If the custom interceptor extends the com.ibm.websphere.security.WebSphereBaseTrustAssociationInterceptor property, implement the following new method to initialize the interceptor:

```
public int init (java.util.Properties props);
```

WebSphere Application Server checks the return status before using the trust association implementation. Zero (0) is the default value for indicating that the interceptor is successfully initialized.

However, if a previous implementation of the trust association interceptor returns a different error status, you can either change your implementation to match the expectations or make one of the following changes:

Method 1:

Add the com.ibm.websphere.security.trustassociation.initStatus property in the trust association interceptor custom properties. Set the property to the value that indicates the interceptor is successfully initialized. All of the other possible values imply failure. In case of failure, the corresponding trust association interceptor is not used.

Method 2:

Add the com.ibm.websphere.security.trustassociation.ignoreInitStatus property in the trust association interceptor custom properties. Set the value of this property to `true`, which tells WebSphere Application Server to ignore the status of this method. If you add this property to the custom properties, WebSphere Application Server does not check the return status, which is similar to previous versions of WebSphere Application Server.

The public int init (java.util.Properties props method replaces the public int init (String propsFile) method.

The init(Properties) method accepts a java.util.Properties object, which contains the set of properties that is required to initialize the interceptor. All of the properties set for an interceptor are sent to this method. The interceptor can then use these properties to initialize itself. For example, in the product-provided implementation for the WebSEAL server, this method reads the hosts and ports so that a request coming in can be verified to come from trusted hosts and ports. A return value of Zero (0) implies that the interceptor initialization is successful. Any other value implies that the initialization is not successful and the interceptor is not used.

The init(String) method still works if you want to use it instead of implementing the init(Properties) method. The only requirement is that you enter the file name containing the custom trust association properties using the **Custom Properties** link of the interceptor in the administrative console or by using scripts. You can enter the property using either of the following methods. The first method is used for backward compatibility with previous versions of WebSphere Application Server.

Method 1:

The same property names used in the previous release are used to obtain the file name. The file name is obtained by concatenating `.config` to the com.ibm.websphere.security.trustassociation.types property value. If the `myTAI.properties` file is located in the *`profile_root`*`/properties` directory, set the following properties:

- com.ibm.websphere.security.trustassociation.types = myTAItype
- com.ibm.websphere.security.trustassociation.myTAItype.config = *profile_root*/properties/myTAI.properties

Method 2:

You can set the com.ibm.websphere.security.trustassociation.initPropsFile property in the trust association custom properties to the location of the file. For example, set the following property:

```
com.ibm.websphere.security.trustassociation.initPropsFile=
profile_root/properties/myTAI.properties
```

The previous line of code is split into two lines for illustrative purposes only. Type as one continuous line.

However, it is highly recommended that your implementation be changed to implement the init(Properties) method instead of relying on the init (String propsfile) method.

Migrating custom trust association interceptors

The trust associations from previous versions of WebSphere Application Server are not automatically migrated to WebSphere Application Server Version 8.0. You can manually migrate these trust associations using the following steps:

Procedure

1. Recompile the implementation file, if necessary.

   For more information, refer to the "Changes to the custom trust association interceptors" section previously discussed in this document.

   a. Enter QSH from a command line to start the QShell environment.

   b. Change to the directory that contains your Java source file.

   c. Enter the command to recompile the implementation file.

```
javac -Djava.version=1.6 -classpath
app_server_root/plugins/com.ibm.ws.runtime.jar:install_root/dev/JavaEE/j2ee.jar your_implementation_file.java
```

2. Copy the custom trust association interceptor class files to a location in your product class path. Copy these class files into the *profile_root*/classes directory.
3. Start WebSphere Application Server.
4. Enable security to use the trust association interceptor. The properties that are located in your custom trust association properties file and in the trustedserver.properties file are not migrated from previous versions of WebSphere Application Server. You must migrate the appropriate properties to WebSphere Application Server Version 8.0 using the trust association panels in the administrative console.

   For more information, see Configuring trust association interceptors.

Migrating Common Object Request Broker Architecture programmatic login to Java Authentication and Authorization Service (CORBA and JAAS)

Use this topic as an example of how to perform programmatic login using the CORBA-based programmatic login APIs.

Before you begin

This document outlines the deprecated Common Object Request Broker Architecture (CORBA) programmatic login APIs and the alternatives that are provided by JAAS. WebSphere Application Server fully supports the Java Authentication and Authorization Service (JAAS) as programmatic login application programming interfaces (API). Refer to the *Securing applications and their environment* PDF for more details on JAAS support.

The following list includes the deprecated CORBA programmatic login APIs.

- `profile_root/installedApps/sampleApp.ear/default_app.war/WEB-INF/classes/ServerSideAuthenticator.java`.
- **org.omg.SecurityLevel2.Credentials**. This API is included with the product, but it is not recommended that you use the API.

The APIs that are provided in WebSphere Application Server are a combination of standard JAAS APIs and a product implementation of standard JAAS interfaces.

The following information is only a summary; refer to the JAAS documentation for your platform located at: http://www.ibm.com/developerworks/java/jdk/security/ .

- Programmatic login APIs:
  - javax.security.auth.login.LoginContext
  - javax.security.auth.callback.CallbackHandler interface: The WebSphere Application Server product provides the following implementation of the javax.security.auth.callback.CallbackHandler interface:
    com.ibm.websphere.security.auth.callback.WSCallbackHandlerImpl
    Provides a non-prompt CallbackHandler handler when the application pushes basic authentication data (user ID, password, and security realm) or token data to product login modules. This API is recommended for server-side login.
    com.ibm.websphere.security.auth.callback.WSGUICallbackHandlerImpl
    Provides a login prompt CallbackHandler handler to gather basic authentication data (user ID, password, and security realm). This API is recommended for client-side login.

    If this API is used on the server side, the server is blocked for input.
  - javax.security.auth.callback.Callback interface:
    javax.security.auth.callback.NameCallback
    Provided by JAAS to pass the user name to the LoginModules interface.
    javax.security.auth.callback.PasswordCallback
    Provided by JAAS to pass the password to the LoginModules interface.
    com.ibm.websphere.security.auth.callback.WSCredTokenCallbackImpl
    Provided by the product to perform a token-based login. With this API, an application can pass a token-byte array to the LoginModules interface.
  - **javax.security.auth.spi.LoginModule interface**

    WebSphere Application Server provides a LoginModules implementation for client and server-side login. Refer to the *Securing applications and their environment* PDF for details.
- javax.security.Subject:
  com.ibm.websphere.security.auth.WSSubject
  An extension provided by the product to invoke remote J2EE resources using the credentials in the javax.security.Subject
  com.ibm.websphere.security.cred.WSCredential
  After a successful JAAS login with the WebSphere Application Server LoginModules interfaces, a com.ibm.websphere.security.cred.WSCredential credential is created and stored in the Subject.
  com.ibm.websphere.security.auth.WSPrincipal
  An authenticated principal that is created and stored in a Subject that is authenticated by the WebSphere Application Server LoginModules interface.

Procedure

1. Use the following as an example of how to perform programmatic login using the CORBA-based programmatic login APIs: The CORBA-based programmatic login APIs are replaced by JAAS login.

   Note: The LoginHelper application programming interface (API) that is used in the following example is deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release. It is recommended that you use the JAAS programmatic login APIs that are shown in the next step.

```
public class TestClient {
...
private void performLogin() {
// Get the ID and password of the user.
String userid = customGetUserid();
String password = customGetPassword();

// Create a new security context to hold authentication data.
LoginHelper loginHelper = new LoginHelper();
try {
// Provide the ID and password of the user for authentication.
org.omg.SecurityLevel2.Credentials credentials =
loginHelper.login(userid, password);

// Use the new credentials for all future invocations.
loginHelper.setInvocationCredentials(credentials);
// Retrieve the name of the user from the credentials
// so we can tell the user that login succeeded.

String username = loginHelper.getUserName(credentials);
System.out.println("Security context set for user: "+username);
} catch (org.omg.SecurityLevel2.LoginFailed e) {
// Handle the LoginFailed exception.
}
}
...
}
```

2. Use the following example to migrate the CORBA-based programmatic login APIs to the JAAS programmatic login APIs.

   The following example assumes that the application code is granted for the required Java 2 security permissions. For more information, see the *Securing applications and their environment* PDF and the JAAS documentation located at http://www.ibm.com/developerworks/java/jdk/security/.

```
public class TestClient {
...
private void performLogin() {
// Create a new JAAS LoginContext.
javax.security.auth.login.LoginContext lc = null;

try {
// Use GUI prompt to gather the BasicAuth data.
lc = new javax.security.auth.login.LoginContext("WSLogin",
new com.ibm.websphere.security.auth.callback.WSGUICallbackHandlerImpl());

// create a LoginContext and specify a CallbackHandler implementation
// CallbackHandler implementation determine how authentication data is collected
// in this case, the authentication date is collected by  login prompt
//   and pass to the authentication mechanism implemented by the LoginModule.
} catch (javax.security.auth.login.LoginException e) {
System.err.println("ERROR: failed to instantiate a LoginContext and the exception: "
+ e.getMessage());
e.printStackTrace();

// may be javax.security.auth.AuthPermission "createLoginContext" is not granted
//   to the application, or the JAAS Login Configuration is not defined.
}

if (lc != null)
try {
lc.login();  // perform login
javax.security.auth.Subject s = lc.getSubject();
// get the authenticated subject

// Invoke a J2EE resources using the authenticated subject
com.ibm.websphere.security.auth.WSSubject.doAs(s,
new java.security.PrivilegedAction() {
public Object run() {
try {
bankAccount.deposit(100.00);  // where bankAccount is an protected EJB
} catch (Exception e) {
System.out.println("ERROR: error while accessing EJB resource, exception: "
+ e.getMessage());
e.printStackTrace();
```

```
}
return null;
}
}
);

// Retrieve the name of the principal from the Subject
// so we can tell the user that login succeeded,
// should only be one WSPrincipal.
java.util.Set ps =
s.getPrincipals(com.ibm.websphere.security.auth.WSPrincipal.class);
java.util.Iterator it = ps.iterator();
while (it.hasNext()) {
com.ibm.websphere.security.auth.WSPrincipal p =
(com.ibm.websphere.security.auth.WSPrincipal) it.next();
System.out.println("Principal: " + p.getName());
}
} catch (javax.security.auth.login.LoginException e) {
System.err.println("ERROR: login failed with exception: " + e.getMessage());
e.printStackTrace();

// login failed, might want to provide relogin logic
}
}
...
}
```

Migrating from the CustomLoginServlet class to servlet filters

Use this topic to allow migration in an application that uses form-based login and servlet filters without the use of the CustomLoginServlet class.

Before you begin

The CustomLoginServlet class is deprecated in WebSphere Application Server Version 5. Those applications using the CustomLoginServlet class to perform authentication now need to use form-based login. Using the form-based login mechanism, you can control the look and feel of the login screen. In form-based login, a login page is specified and displays when retrieving the user ID and password information. You also can specify an error page that displays when authentication fails.

If login and error pages are not enough to implement the CustomLoginServlet class, use servlet filters. Servlet filters can dynamically intercept requests and responses to transform or use the information that is contained in the requests or responses. One or more servlet filters attach to a servlet or a group of servlets. Servlet filters also can attach to JavaServer Pages (JSP) files and HTML pages. All the attached servlet filters are called before invoking the servlet.

Both form-based login and servlet filters are supported by any Servlet 2.3 specification-compliant web container. A form login servlet performs the authentication and servlet filters can perform additional authentication, auditing, or logging tasks.

To perform pre-login and post-login actions using servlet filters, configure these servlet filters for either form login page or for /j_security_check URL. The j_security_check is posted by the form login page with the j_username parameter that contains the user name and the j_password parameter that contains the password. A servlet filter can use user name and password information to perform more authentication or meet other special needs.

Procedure

1. Develop a form login page and error page for the application.
   Refer to the *Securing applications and their environment* PDF for details.
2. Configure the form login page and the error page for the application.
   Refer to the *Securing applications and their environment* PDF for details.
3. Develop servlet filters if additional processing is required before and after form login authentication.

Refer to the *Securing applications and their environment* PDF for details.

4. Configure the servlet filters that are developed in the previous step for either the form login page URL or for the /j_security_check URL. Use an assembly tool or development tools like Rational Application Developer to configure filters. After configuring the servlet filters, the `web-xml` file contains two stanzas. The first stanza contains the servlet filter configuration, the servlet filter, and its implementation class. The second stanza contains the filter mapping section and a mapping of the servlet filter to the URL.

   For more information, see the *Securing applications and their environment* PDF.

Results

This migration results in an application that uses form-based login and servlet filters without the use of the CustomLoginServlet class.

What to do next

The new application uses form-based login and servlet filters to replace the CustomLoginServlet class. Servlet filters also are used to perform additional authentication, auditing, and logging.

Migrating Java 2 security policy

Use this topic for guidance pertaining to migrating Java 2 security policy.

About this task

Previous WebSphere Application Server releases

WebSphere Application Server uses the Java 2 security manager in the server runtime to prevent enterprise applications from calling the System.exit and the System.setSecurityManager methods. These two Java application programming interfaces (API) have undesirable consequences if called by enterprise applications. The System.exit API, for example, causes the Java virtual machine (application server process) to exit prematurely, which is not a beneficial operation for an application server.

To support Java 2 security properly, all the server runtime must be marked as `privileged` (with doPrivileged API calls inserted in the correct places), and identify the default permission sets or policy. Application code is not privileged and subject to the permissions that are defined in the policy files. The doPrivileged instrumentation is important and necessary to support Java 2 security. Without it, the application code must be granted the permissions that are required by the server runtime. This situation is due to the design and algorithm that is used by Java 2 security to enforce permission checks. Refer to the Java 2 security check permission algorithm.

The following two permissions are enforced by the Java 2 security manager (hard coded) for WebSphere Application Server:

- java.lang.RuntimePermission(exitVM)
- java.lang.RuntimePermission(setSecurityManager)

Application code is denied access to these permissions regardless of what is in the Java 2 security policy. However, the server runtime is granted these permissions. All the other permission checks are not enforced.

Only two permissions are supported:

- java.net.SocketPermission
- java.net.NetPermission

However, not all the product server runtime is properly marked as privileged. You must grant the application code all the other permissions besides the two listed previously or the enterprise application can potentially fail to run. This Java 2 security policy for enterprise applications is liberal.

What changed

Java 2 Security is fully supported in WebSphere Application Server, which means that all permissions are enforced. The default Java 2 security policy for an enterprise application is the recommended permission set defined by the Java Platform, Enterprise Edition (Java EE) Version 1.4 specification. Refer to the *profile_root*/config/cells/*cell_name*/nodes/*node_name*/app.policy file for the default Java 2 security policy that is granted to enterprise applications. This policy is a much more stringent compared to previous releases.

All policy is declarative. The product security manager honors all policy that is declared in the policy files. There is an exception to this rule: enterprise applications are denied access to permissions that are declared in the *profile_root*/config/cells/*cell_name*/filter.policy file.

Note: The default Java 2 security policy for enterprise applications is much more stringent and all the permissions are enforced in WebSphere Application Server Version 8.0. The security policy might fail because the application code does not have the necessary permissions granted where system resources, such as file I/O, can be programmatically accessed and are now subject to the permission checking.

In application code, do not use the setSecurityManager permission to set a security manager. When an application uses the setSecurityManager permission, there is a conflict with the internal security manager within WebSphere Application Server. If you must set a security manager in an application for RMI purposes, you also must enable the **Use Java 2 security to restrict application access to local resources** option on the Global security page within the WebSphere Application Server administrative console. WebSphere Application Server then registers a security manager. The application code can verify that this security manager is registered by using System.getSecurityManager() application programming interface (API).

Migrating system properties

The following system properties are used in previous releases in relation to Java 2 security:

- **java.security.policy**. The absolute path of the policy file (action required). This system property contains both system permissions (permissions granted to the Java virtual machine (JVM) and the product server runtime) and enterprise application permissions. Migrate the Java 2 security policy of the enterprise application to Version 8.0. For Java 2 security policy migration, see the steps for migrating Java 2 security policy.
- **enableJava2Security**. Used to enable Java 2 security enforcement (no action required). This system property is deprecated; a flag in the WebSphere configuration application programming interface (API) is used to control whether to enable Java 2 security. Enable this option through the administrative console.
- **was.home**. Expanded to the installation directory of WebSphere Application Server (action might be required). This system property is deprecated; superseded by the ${user.install.root} and ${was.install.root} properties. If the directory contains instance-specific data then ${user.install.root} is used; otherwise ${was.install.root} is used. Use these properties interchangeably for the WebSphere Application Server or the WebSphere Application Server, Network Deployment environments. See the steps for migrating Java 2 security policy.

Migrating the Java 2 Security Policy

No easy way exists to migrate the Java policy file to Version 8.0 automatically because of a mixture of system permissions and application permissions in the same policy file. Manually copy the Java 2 security policy for enterprise applications to a was.policy or app.policy file. However, migrating the Java 2 security policy to a was.policy file is preferable because symbols or relative code base is used instead of an absolute code base. This process has many advantages. Grant the permissions that are defined in the was.policy to the specific enterprise application only, while permissions in the app.policy file apply to all the enterprise applications that run on the node where the app.policy file belongs.

Refer to the *Securing applications and their environment* PDF for more details on policy management.

The following example illustrates the migration of a Java 2 security policy from a previous release. The contents include the Java 2 security policy file for the `app1.ear` enterprise application and the system permissions, which are permissions that are granted to the Java virtual machine (JVM) and the product server runtime.

The default location for the Java 2 security policy file is *profile_root*`/properties/java.policy`. Default permissions are omitted for clarity:

```
// For product Samples
   grant codeBase "file:${app_server_root}/installedApps/app1.ear/-" {
     permission java.security.SecurityPermission "printIdentity";
     permission java.io.FilePermission "${app_server_root}${/}temp${/}somefile.txt",
       "read";
   };
```

For clarity of illustration, all the permissions are migrated as the application level permissions in this example. However, you can grant permissions at a more granular level at the component level (Web, enterprise beans, connector or utility Java archive (JAR) component level) or you can grant permissions to a particular component.

Procedure

1. Ensure that Java 2 security is disabled on the application server.
2. Create a new `was.policy` file, if the file is not present, or update the `was.policy` file for migrated applications in the configuration repository with the following contents:

   ```
   grant codeBase "file:${application}" {
        permission java.security.SecurityPermission "printIdentity";
        permission java.io.FilePermission "
                ${user.install.root}${/}temp${/}somefile.txt", "read";
      };
   ```

   The third and fourth lines in the previous code sample are presented on two lines for illustrative purposes only.

   The `was.policy` file is located in the *profile_root*`/config/cells/`*cell_name*`/applications/app.ear/deployments/app/META-INF/` directory.
3. Use an assembly tool to attach the `was.policy` file to the enterprise archive (EAR) file.

   You also can use an assembly tool to validate the contents of the `was.policy` file. For more information, see the *Securing applications and their environment* PDF.
4. Validate that the enterprise application does not require additional permissions to the migrated Java 2 security permissions and the default permissions set declared in the `${user.install.root}/config/cells/`*cell_name*`/nodes/`*node_name*`/app.policy` file. This validation requires code review, code inspection, application documentation review, and sandbox testing of migrated enterprise applications with Java 2 security enabled in a preproduction environment. Refer to developer kit APIs protected by Java 2 security for information about which APIs are protected by Java 2 security. If you use third-party libraries, consult the vendor documentation for APIs that are protected by Java 2 security. Verify that the application is granted all the required permissions, or it might fail to run when Java 2 security is enabled.
5. Perform preproduction testing of the migrated enterprise application with Java 2 security enabled. Enable trace for the WebSphere Application Server Java 2 security manager in a preproduction testing environment with the following trace string: `com.ibm.ws.security.core.SecurityManager=all=enabled`. This trace function can be helpful in debugging the AccessControlException exception that is created when an application is not granted the required permission or some system code is not properly marked as privileged. The trace dumps the stack trace and permissions that are granted to the classes on the call stack when the exception is created.

   For more information, see the *Securing applications and their environment* PDF.

Note: Because the Java 2 security policy is much more stringent compared with previous releases, the administrator or deployer must review their enterprise applications to see if extra permissions are required before enabling Java 2 security. If the enterprise applications are not granted the required permissions, they fail to run.

Migrating with Tivoli Access Manager for authentication enabled

When Tivoli Access Manager security is configured for your existing environment and security is enabled, you can migrate to WebSphere Application Server, Version 8.0.

Before you begin

Your profiles must be migrated using the migration tools to migrate product configurations.

Important: Do not restart the WebSphere Application Server Version 8.0 server until after performing the following procedure. The migration tools omit some files that enable the server to start correctly.

About this task

After migrating your profiles, additional steps are required when Tivoli Access Manager security is configured.

Note: WebSphere Application Server Version 8.0 hosts Tivoli Access Manager specific files under the `%WAS_HOME%/tivoli/tam` directory. In previous versions, these files were hosted under the `%WAS_HOME%/java/jre/` hierarchy.

Procedure

1. Copy the `profile_root1/PolicyDirector` directory and it's contents to `profile_root2/PolicyDirector`. For this example:
   - *profile_root1* is the root directory of the profile being migrated.
   - *profile_root2* is the root directory of the version 6.1 profile.
   a. From an IBM i command line, type **STRQSH** and press **Enter**.
   b. Type `cp -R profile_root1/PolicyDirector profile_root2` and press **Enter**.
2. Copy the key file of the profile being migrated to the version 8.0 profile. The location of the key file is defined in `profile_root1/PolicyDirector/PdPerm.properties`. For this example:
   - The `PdPerm.properties` file contains *pdcert-url=file\:/QIBM/UserData/WebAS51/Base/AppSvr1/etc/AppSvr1.kdb*.
   - `/QIBM/UserData/WebAS51/Base/AppSvr1` is the root directory of a Version 6.1 profile.
   a. From an IBM i command line type **STRQSH** and press **Enter**.
   b. Type `cp /QIBM/UserData/WebAS51/Base/AppSvr1/etc/AppSvr1.kdb profile_root2/etc/AppSvr1.kdb` and press **Enter**.
3. Edit the property values in `profile_root2/PolicyDirector/PdPerm.properties` and in `profile_root2/PolicyDirector/Pd.properties` to replace occurrences of *profile_root1* with *profile_root2* in the file path name values.

Migrating Java thin clients that use the password encoding algorithm

To migrate Java thin clients that are enabled for OS400 password encoding, use the following information to modify the Java client invocation so that the os400.security.password properties are no longer set on the invocation.

About this task

The password encoding feature offers the following encoding algorithms:

- XOR, which is the default
- OS400

In Version 5 and later, the value of the os400.security.password.validation.list.object property is dependant upon the property value passed to the thin client using the JAVA_FLAGS environment variable. The JAVA_FLAGS environment variable is set by the **setupClient** script. The **setupClient** script calls the **setupCmdLine** script, which is where the value for the os400.security.password.validation.list.object property is set. For example, if a Version 6.*x* Base Edition Java client is passed **-profileName default**, then the **setupClient** script calls the *profile_root*/default/bin/setupCmdLine file.

To migrate Java thin clients that are enabled for OS400 password encoding, modify the Java client invocation so that the os400.security.password properties are no longer set on the invocation. The following code sample does not contain the os400.security.password properties:

```
java -classpath $MY_CLIENT_CLASSES:app_server_root/classes/wsa400.jar:$WAS_CLASSPATH \
  $CLIENTSAS $JAVA_FLAGS \
  -Djava.naming.factory.initial=com.ibm.websphere.naming.WsnInitialContextFactory \
  -Djava.naming.provider.url=iiop://server1:10151 \
  MyClientClass $*
```

Perform the following steps if the following condition is true:

- If the passwords in the sas.client.props file for that profile are encoded with the OS400 password encoding algorithm

Procedure

1. Replace all of the OS400 encoded passwords, which have {OS400}) prefixes in the sas.client.props file for the Application Server profile, with the clear text values of the passwords.
2. Encode the passwords using the **PropFilePasswordEncoder** Qshell command.

   For more information, see PropFilePasswordEncoder command reference.

Results

Attention: You can configure a WebSphere Application Server profile to encode passwords with the XOR algorithm even though the profile is enabled to decode passwords that were encoded with either the OS400 algorithm or the XOR algorithm. If you encode these passwords with the XOR algorithm, then the passwords in the sas.client.props file are encoded with the XOR algorithm.

Enabling security

The following provides information on how to configure security when security was not enabled during the WebSphere Application Sever profile creation.

Before you begin

When you are installing WebSphere Application Server, it is recommended that you install with security enabled. By design, this option ensures that everything has been properly configured. By enabling security, you protect your server from unauthorized users and are then able to provide application isolation and requirements for authenticating application users.

It is helpful to understand security from an infrastructure perspective so that you know the advantages of different authentication mechanisms, user registries, authentication protocols, and so on. Picking the right security components to meet your needs is a part of configuring security. The following sections help you make these decisions.

Read the following articles before continuing with the security configuration:
- Server and administrative security
- Security

After you understand the security components, you can proceed to configure security in WebSphere Application Server.

Procedure

1. Start the WebSphere Application Server administrative console.

   If security is currently disabled, you are prompted for a user ID. Log in with any user ID. However, if security is currently enabled, you are prompted for both a user ID and a password. Log in with a predefined administrative user ID and password.

2. Click **Security > Global security**.

   Use the Security Configuration Wizard, or configure security manually. The configuration order is not important.

   gotcha: You must separately enable administrative security, and application security. Because of this split, WebSphere Application Server clients must know whether application security is disabled at the target server. Administrative security is enabled, by default. Application security is disabled, by default. Before you attempt to enable application security on the target server, verify that administrative security is enabled on that server. Application security can be in effect only when administrative security is enabled.

   For more information on manual configuration, see “Authenticating users” on page 1254.

3. Configure the user account repository. For more information, see “Selecting a registry or repository” on page 1254. On the Global security panel, you can configure user account repositories such as federated repositories, local operating system, stand-alone Lightweight Directory Access Protocol (LDAP) registry, and stand-alone custom registry.

   Note: You can choose to specify either a server ID and password for interoperability or enable a WebSphere Application Server installation to automatically generate an internal server ID. For more information about automatically generating server IDs, see “Local operating system settings” on page 1259.

   One of the details common to all user registries or repositories is the *Primary administrative user name*. This ID is a member of the chosen repository, but also has special privileges in WebSphere Application Server. The privileges for this ID and the privileges that are associated with the administrative role ID are the same. The Primary administrative user name can access all of the protected administrative methods.

   In stand-alone LDAP registries, verify that the Primary administrative user name is a member of the repository and not just the LDAP administrative role ID. The entry must be searchable.

   The Primary administrative user name does **not** run WebSphere Application Server processes. Rather, the process ID runs the WebSphere Application Server processes.

   In the default configuration, WebSphere Application Server processes run under the QEJBSVR system-provided user profile.

4. Select the **Set as current** option after you configure the user account repository. When you click **Apply** and the Enable administrative security option is set, a verification occurs to see if an administrative user ID has been configured and is present in the active user registry. The administrative user ID can be specified at the active user registry panel or from the console users link. If you do not configure an administrative ID for the active user registry, the validation fails.

   Note: When you switch user registries, the `admin-authz.xml` file should be cleared of existing administrative ids and application names. Exceptions will occur in the logs for ids that exist in the `admin-authz.xml` file but do not exist in the current user registry.

5. Configure the authentication mechanism.

Configure Lightweight Third-Party Authentication (LTPA) or Kerberos, which is new to this release of WebSphere Application Server, under Authentication mechanisms and expiration. LTPA credentials can be forwarded to other machines. For security reasons, credential expire; however, you can configure the expiration dates on the console. LTPA credentials enable browsers to visit different product servers, which means you do not have to authenticate multiple times. For more information, see Configuring the Lightweight Third Party Authentication mechanism

Note: You can configure Simple WebSphere Authentication Mechanism (SWAM) as your authentication mechanism. However, SWAM was deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release. SWAM credentials are not forwardable to other machines and for that reason do not expire.

6. Optional: Import and export the LTPA keys for cross-cell single Sign-on (SSO) between cells. For more information, see the following articles:
   - Exporting Lightweight Third Party Authentication keys.
   - Importing Lightweight Third Party Authentication keys

   gotcha: If one of the cells you are connecting to resides on a Version 6.0.x system, see the topic Configuring Lightweight Third Party Authentication keys in the Version 6.0.x Information Center for more information.
7. Configure the authentication protocol for special security requirements from Java clients, if needed.

   You can configure Common Secure Interoperability Version 2 (CSIv2) through links on the Global security panel. The Security Authentication Service (SAS) protocol is provided for backwards compatibility with previous product releases, but is deprecated. Links to the SAS protocol panels display on the Global security panel if your environment contains servers that use previous versions of WebSphere Application Server and support the SAS protocol. For details on configuring CSIv2 or SAS, see the article, “Configuring Common Secure Interoperability Version 2 (CSIV2) inbound and outbound communication settings” on page 1561.

   Attention: IBM no longer ships or supports the Secure Authentication Service (SAS) IIOP security protocol. It is recommended that you use the Common Secure Interoperability version 2 (CSIv2) protocol.
8. Click **Security > Global security** to configure the rest of the security settings and enable security. For information about these settings, see “Global security settings” on page 1187.
9. Validate the completed security configuration by clicking **OK** or **Apply**. If problems occur, they display at the top of the console page in red type.
10. If there are no validation problems, click **Save** to save the settings to a file that the server uses when it restarts. Saving writes the settings to the configuration repository.

    Important: If you do not click **Apply** or **OK** in the Global security panel before you click **Save**, your changes are not written to the repository. The server must be restarted for any changes to take effect when you start the administrative console.
11. Start the WebSphere Application Server administrative console.

    If security is currently disabled, log in with any user ID. If security is currently enabled, log in with a predefined administrative ID and password. This ID is typically the server user ID that is specified when you configured the user registry.

Administrative security

Administrative security determines whether security is used at all, the type of registry against which authentication takes place, and other values, many of which act as defaults. Proper planning is required because incorrectly enabling administrative security can lock you out of the administrative console or cause the server to end abnormally.

Note: It is strongly recommended that you allow the default installation to install administrative security as on by default.

Administrative security can be thought of as a "big switch" that activates a wide variety of security settings for WebSphere Application Server. Values for these settings can be specified, but they will not take effect until administrative security is activated. The settings include the authentication of users, the use of Secure Sockets Layer (SSL), and the choice of user account repository. In particular, application security, including authentication and role-based authorization, is not enforced unless administrative security is active. Administrative security is enabled by default.

Note: Administrative security need not be activated in order for WebSphere applications to make use of JSSE methods to encrypt communication to remote sites.

Administrative security represents the security configuration that is effective for the entire security domain. A *security domain* consists of all of the servers that are configured with the same user registry *realm* name. In some cases, the realm can be the machine name of a local operating system registry. In this case, all of the application servers must reside on the same physical machine. In other cases, the realm can be the machine name of a stand-alone Lightweight Directory Access Protocol (LDAP) registry.

The basic requirement for a security domain is that the access ID that is returned by the registry or repository from one server within the security domain is the same access ID as that returned from the registry or repository on any other server within the same security domain. The *access ID* is the unique identification of a user and is used during authorization to determine if access is permitted to the resource.

The administrative security configuration applies to every server within the security domain.

Why turn on administrative security?

Turning on administrative security activates the settings that protect your server from unauthorized users. Administrative security is enabled by default during the profile creation time. There might be some environments where no security is needed such as a development system. On these systems you can elect to disable administrative security. However, in most environments you should keep unauthorized users from accessing the administrative console and your business applications. Administrative security must be enabled to restrict access.

What does administrative security protect?

The configuration of administrative security for a security domain involves configuring the following technologies:

- Authentication of HTTP clients
- Authentication of IIOP clients
- Administrative console security
- Naming security
- Use of SSL transports
- Role-based authorization checks of servlets, enterprise beans, and mbeans
- Propagation of identities (RunAs)
- The common user registry
- The authentication mechanism
- Other security information that defines the behavior of a security domain includes:
  - The authentication protocol (Remote Method Invocation over the Internet Inter-ORB Protocol (RMI/IIOP) security)
  - Other miscellaneous attributes

Note: It is recommended that before registering a node with an administrative agent process, that you first have administrative security enabled in the administrative agent and base profile. Once you register a profile with the administrative agent, the state of administrative security enablement cannot be changed.

Application security

Application security enables security for the applications in your environment. This type of security provides application isolation and requirements for authenticating application users

In previous releases of WebSphere Application Server, when a user enabled global security, both administrative and application security were enabled. In WebSphere Application Server Version 6.1, the previous notion of global security is split into administrative security and application security, each of which you can enable separately.

As a result of this split, WebSphere Application Server clients must know whether application security is disabled at the target server. Administrative security is enabled, by default. Application security is disabled, by default. Before you can enable application security, you must verify that administrative security is enabled. Application security is in effect only when administrative security is enabled.

An Application Server Enablement Tag, which is specific to WebSphere Application Server, is imported into the Interoperable Object Reference (IOR) to indicate if application security is disabled for the server where the object lives. This tag is server-specific and enables clients to know when application security is disabled at the target server of its request.

For web resources, when application security is enabled, security constraints on those resources in web.xml are enforced. When accessing a protected resource, a web client is prompted for authentication.

For enterprise bean resources, when application security is disabled, the client Common Secure Interoperability version 2 (CSIv2) code ignores the CSIv2 security tags for objects that are unknown system objects. When pure clients see that application security is disabled, these clients prompt for naming lookups, but do not prompt for enterprise bean operations.

Java 2 security

Java 2 security provides a policy-based, fine-grain access control mechanism that increases overall system integrity by checking for permissions before allowing access to certain protected system resources. Java 2 security guards access to system resources such as file I/O, sockets, and properties. Java 2 Platform, Enterprise Edition (J2EE) security guards access to web resources such as servlets, JavaServer Pages (JSP) files and Enterprise JavaBeans (EJB) methods.

Because Java 2 security is relatively new, many existing or even new applications might not be prepared for the very fine-grain access control programming model that Java 2 security is capable of enforcing. Administrators need to understand the possible consequences of enabling Java 2 security if applications are not prepared for Java 2 security. Java 2 security places some new requirements on application developers and administrators.

Important: Java 2 security only restricts Java programs that run in a Java virtual machine that has Java 2 security enabled. It does not protect system resources if Java 2 Security is disabled or if system resources are accessed from other programs or commands. Therefore, if you want to protect your system resources, you need to use operating system security.

Note: The application server does not support a custom Java security manager implementation.

Java 2 security for deployers and administrators

Although Java 2 security is supported, it is disabled by default. You can configure Java 2 security and administrative security independently of one another. Disabling administrative security does not disable Java 2 security automatically. You need to explicitly disable it.

If your applications, or third-party libraries are not ready, having Java 2 security enabled causes problems. You can identify these problems as Java 2 security AccessControlExceptions in the system log or trace files. If you are unsure about the Java 2 security readiness of your applications, disable Java 2 security initially to get your application installed and verify that it is working properly.

The policy embodied by these policy files cannot be made more restrictive because the product might not have the necessary Java 2 security doPrivileged APIs in place. The restrictive policy is the default policy. You can grant additional permissions, but you cannot make the default more restrictive because AccessControlExceptions exceptions are generated from within WebSphere Application Server. The product does not support a more restrictive policy than the default that is defined in the policy files previously mentioned.

Several policy files are used to define the security policy for the Java process. These policy files are static (code base is defined in the policy file) and in the default policy format provided by the IBM Developer Kit, Java Technology Edition. For enterprise application resources and utility libraries, WebSphere Application Server provides dynamic policy support. The code base is dynamically calculated based on deployment information and permissions are granted based on template policy files during runtime. Refer to the “Java 2 security policy files” on page 1181 for more information.

Syntax errors in the policy files cause the application server process to fail, so edit these policy files carefully.

Note: Edit these policy files using the Policy Tool that is provided by the IBM Developer Kit, Java Technology Edition. See Using PolicyTool to edit policy files for Java 2 security for more information.

If an application is not prepared for Java 2 security, if the application provider does not provide a `was.policy` file as part of the application, or if the application provider does not communicate the expected permissions the application is likely to cause Java 2 security access control exceptions at runtime. It might not be obvious that an application is not prepared for Java 2 security. Several run-time debugging aids help troubleshoot applications that might have access control exceptions. See the Java 2 security debugging aids for more details. See “Handling applications that are not Java 2 security ready” on page 1179 for information and strategies for dealing with such applications.

It is important to note when Java Security is enabled in the administrative security settings, the installed security manager does not currently check modifyThread and modifyThreadGroup permissions for non-system threads. Allowing web and Enterprise JavaBeans (EJB) application code to create or modify a thread can have a negative impact on other components of the container and can affect the capability of the container to manage enterprise bean life cycles and transactions.

Java 2 security for application developers

Application developers must understand the permissions that are granted in the default WebSphere policy and the permission requirements of the SDK APIs that their application calls to know whether additional permissions are required. The Permissions in the Java 2 SDK reference in the resources section describes which APIs require which permission.

Application providers can assume that applications have the permissions granted in the default policy previously mentioned. Applications that access resources not covered by the default WebSphere policy are required to grant the additional Java 2 security permissions to the application.

While it is possible to grant the application additional permissions in one of the other dynamic WebSphere policy files or in one of the more traditional `java.policy` static policy files, the `was.policy` file, which is embedded in the EAR file ensures the additional permissions are scoped to the exact application that requires them. Scoping the permission beyond the application code that requires it can permit code that normally does not have permission to access particular resources.

If an application component is being developed, like a library that might actually be included in more than one `.ear` file, then the library developer needs to document the required Java 2 permissions that are required by the application assembler. There is no `was.policy` file for library-type components. The developer must communicate the required permissions through application programming interface (API) documentation or some other external documentation.

If the component library is shared by multiple enterprise applications, the permissions can be granted to all enterprise applications on the node in the `app.policy` file.

Note: Updates to the `app.policy` file only apply to the enterprise applications on the node to which the `app.policy` file belongs.

If the permission is only used internally by the component library and the application is never granted access to resources that are protected by the permission, it might be necessary to mark the code as **privileged**. Refer to the, AccessControlException, topic for more details. However, improperly inserting a doPrivileged call might open up security holes. Understand the implication of doPrivileged call to make a correct judgement.

The section on Dynamic policy files in “Java 2 security policy files” on page 1181 describes how the permissions in the `was.policy` files are granted at runtime.

Developing an application to use with Java 2 security might be a new skill and impose a security awareness not previously required of application developers. Describing the Java 2 security model and the implications on application development is beyond the scope of this section. The following URL can help you get started: http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html.

Debugging Aids

The WebSphere Application Server SYSOUT file and the com.ibm.websphere.java2secman.norethrow property are the two primary aids for debugging.

The WebSphere System Log or Trace Files

The AccessControl exception that is logged in the system log or trace files contains the permission violation that causes the exception, the exception call stack, and the permissions granted to each stack frame. This information is usually enough to determine the missing permission and the code requiring the permission.

The com.ibm.websphere.java2secman.norethrow property

When Java 2 security is enabled in WebSphere Application Server, the security manager component creates a java.security.AccessControl exception when a permission violation occurs. This exception, if not handled, often causes a run-time failure. This exception is also logged in the SYSOUT file.

However, when the Java virtual machine com.ibm.websphere.java2secman.norethrow property is set and has a value of `true`, the security manager does not create the AccessControl exception. This information is logged.

This property is intended for a sandbox or debug environment because it instructs the security manager not to create the AccessControl exception. Java 2 security is not enforced. Do not use this property in a production environment where a relaxed Java 2 security environment weakens the integrity that Java 2 security is intended to produce.

This property is valuable in a sandbox or test environment where the application can be thoroughly tested and where the system log or trace files can be inspected for AccessControl exceptions. Because this property does not create the AccessControl exception, it does not propagate the call stack and does not cause a failure. Without this property, you have to find and fix AccessControl exceptions one at a time.

Handling applications that are not Java 2 security ready

If the increased system integrity that Java 2 security provides is important, then contact the application provider to have the application support Java 2 security or at least communicate the required additional permissions beyond the default WebSphere Application Server policy that must be granted.

The easiest way to deal with such applications is to disable Java 2 security in WebSphere Application Server. The downside is that this solution applies to the entire system and the integrity of the system is not as strong as it might be. Disabling Java 2 security might not be acceptable depending on the organization security policies or risk tolerances.

Another approach is to leave Java 2 security enabled, but to grant either just enough additional permissions or grant all permissions to just the problematic application. Granting permissions however, might not be a trivial thing to do. If the application provider has not communicated the required permissions in some way, no easy way exists to determine what the required permissions are and granting all permissions might be the only choice. You minimize this risk by locating this application on a different node, which might help isolate it from certain resources. Grant the java.security.AllPermission permission in the `was.policy` file that is embedded in the application `.ear` file, for example:

```
grant codeBase "file:${application}" {
            permission java.security.AllPermission;
       };
```

The server.policy file

The `server.policy` file is located in the *profile_root*`/properties` directory.

This policy defines the policy for the WebSphere Application Server classes. At present, all the server processes on the same installation share the same `server.policy` file. However, you can configure this file so that each server process can have a separate `server.policy` file. Define the policy file as the value of the java.security.policy Java system properties . For details of how to define Java system properties, refer to the Process definition section of the Manage application servers file.

The `server.policy` file is not a configuration file managed by the repository and the file replication service. Changes to this file are local and do not get replicated to other machines. Use the `server.policy` file to define Java 2 security policy for server resources. Use the `app.policy` file (per node) or the `was.policy` file (per enterprise application) to define Java 2 security policy for enterprise application resources.

Note: Updates to the `app.policy` file only apply to the enterprise applications on the node to which the app.policy file belongs.

The java.policy file

The file represents the default permissions that are granted to all classes. The policy of this file applies to all the processes launched by the Java Virtual Machine in the WebSphere Application Server.

The java.policy file is located in the${java.home}/lib/security/ directory where ${java.home} is the path to the Software Development Kit (SDK) that you are using. The policy file is used throughout the operating system. Do not edit the java.policy file.

Troubleshooting

Error message CWSCJ0314E

Symptom:

Error message CWSCJ0314E: Current Java 2 security policy reported a potential violation of Java 2 security permission. Refer to Problem Determination Guide for further information.{0}Permission\ :{1}Code\:{2}{3}Stack Trace\:{4}Code Base Location\:{5} Current Java 2 security policy reported a potential violation of Java 2 Security Permission. Refer to Problem Determination Guide for further information.{0}Permission\:{1}Code\:{2}{3}Stack Trace\:{4}Code Base Location\:{5}

Problem:

The Java security manager checkPermission method reported a security exception on the subject permission with debugging information. The reported information can be different with respect to the system configuration. This report is enabled by either configuring a Reliability Availability Service Ability (RAS) trace into debug mode or specifying a Java property.

See Enabling trace for information on how to configure RAS trace in debug mode.

Specify the following property in the JVM Settings panel from the administrative console:
java.security.debug. Valid values include:

access
: Print all debug information including: required permission, code, stack, and code base location.

stack Print debug information including: required permission, code, and stack.

failure Print debug information including: required permission and code.

Recommended response:

The reported exception might be critical to the secure system. Turn on security trace to determine the potential code that might have violated the security policy. After the violating code is determined, verify if the attempted operation is permitted with respect to Java 2 security, by examining all applicable Java 2 security policy files and the application code.

If the application is running with Java Mail, this message might be benign. You can update the was.policy file to grant the following permissions to the application:

```
permission  java.io.FilePermission  "${user.home}${/}.mailcap",  "read";
permission  java.io.FilePermission  "${user.home}${/}.mime.types",  "read";
permission  java.io.FilePermission  "${java.home}${/}lib${/}mailcap",  "read";
permission  java.io.FilePermission  "${java.home}${/}lib${/}mime.types",  "read";
```

SecurityException - Access denied

Symptom:

If Java security is enabled, and permissions to read the jaxm.properties file is not granted, when a SOAPFactory instance is created through a call to javax.xml.soap.SOAPFactory.newInstance(), or a MessageFactory instance is created through a call to MessageFactory.newInstance(), a SecurityException exception occurs, and the following exception is written to the system log:

```
Permission:

      /opt/IBM/WebSphere/AppServer/java/jre/lib/jaxm.properties : access denied
(java.io.FilePermission /opt/IBM/WebSphere/AppServer/java/jre/lib/jaxm.properties
read)

Code:

     com.ibm.ws.wsfvt.test.binding.addr1.binder.AddressBinder
in  {file:/opt/IBM/WebSphere/AppServer/profiles/AppSrv01/installedApps/
```

```
ahp6405Node01Cell/DataBinding.ear/address1.war/WEB-INF/lib
/addressbinder1.jar}

Stack Trace:

java.security.AccessControlException: access denied (java.io.FilePermission
/opt/IBM/WebSphere/AppServer/java/jre/lib/jaxm.properties read)
.
```

Problem:

The Java 2 Security policy reports a potential violation of Java 2 Security permission.

Recommended response:

The SOAPFactory ignores the exception, and continues on to the next means of determining which implementation to load. Therefore, you can ignore the log entry for this security exception.

Because this product uses the SOAPFactory to support other web services technologies, such as WS-Addressing (WS-A), WS-Atomic Transaction (WS-AT), and WS-Notification, you can ignore this SecurityException in any web services application where Java security is enabled.

Messages

Message:	CWSCJ0313E: Java 2 security manager debug message flags are initialized\: TrDebug: {0}, Access: {1}, Stack: {2}, Failure: {3}
Problem:	Configured values of the valid debug message flags for security manager.

Message:	CWSCJ0307E: Unexpected exception is caught when trying to determine the code base location. Exception: {0}
Problem:	An unexpected exception is caught when the code base location is determined.

Java 2 security policy files:

The Java 2 Platform, Enterprise Edition (J2EE) Version 1.3 and later specifications have a well-defined programming model of responsibilities between the container providers and the application code. Using Java 2 security manager to help enforce this programming model is recommended. Certain operations are not supported in the application code because such operations interfere with the behavior and operation of the containers. The Java 2 security manager is used in the product to enforce responsibilities of the container and the application code.

Note: The application server does not support a custom Java security manager implementation.

This product provides support for policy file management. A number of policy files in the product are either static or dynamic. *Dynamic policy* is a template of permissions for a particular type of resource. No relative code base is defined in the dynamic policy template. The code base is dynamically calculated from the deployment and run-time data.

Static policy files

Table 76. Static policy files.

This table lists the location of the static policy files.

Policy file	Location
java.policy	
server.policy	*profile_root*/properties/server.policy. Default permissions are granted to all the product servers.

Table 76. Static policy files (continued).

This table lists the location of the static policy files.

Policy file	Location
client.policy	*profile_root*/properties/client.policy. Default permissions are granted for all of the product client containers and applets on a node.

The static policy files are not managed by configuration and file replication services. Changes made in these files are local and are not replicated to other nodes in the WebSphere Application Server, Network Deployment cell.

Dynamic policy files

Table 77. Dynamic policy files.

This table lists the location of the dynamic policy files.

Policy file	Location
spi.policy	*profile_root*/config/cells/*cell_name*/nodes/*node_name*/spi.policy This template is for the Service Provider Interface (SPI) or the third-party resources that are embedded in the product. Examples of SPI are the Java Message Service (JMS) in MQ Series and Java database connectivity (JDBC) drivers. The code base for the embedded resources are dynamically determined from the configuration (resources.xml file) and run-time data, and permissions that are defined in the spi.policy files are automatically applied to these resources and JAR files that are specified in the class path of a resource adapter. The default permission of the spi.policy file is java.security.AllPermissions.
library.policy	*profile_root*/config/cells/*cell_name*/nodes/*node_name*/library.policy This template is for the library (Java library classes). You can define a shared library to use in multiple product applications. The default permission of the library.policy file is empty.
app.policy	*profile_root*/config/cells/*cell_name*/nodes/*node_name*/app.policy The app.policy file defines the default permissions that are granted to all of the enterprise applications running on *node_name* in *cell_name*. **Note:** Updates to the app.policy file only apply to the enterprise applications on the node to which the app.policy file belongs.
was.policy	*profile_root*/config/cells/*cell_name*/applications/*ear_file_name*/deployments/*application_name*/META-INF/was.policy This template is for application-specific permissions. The was.policy file is embedded in the enterprise archive (EAR) file.
ra.xml	*rar_file_name*/META-INF/was.policy.RAR. This file can have a permission specification that is defined in the ra.xml file. The ra.xml file is embedded in the RAR file.

Grant entries that are specified in the app.policy and was.policy files must have a code base defined. If grant entries are specified without a code base, the policy files are not loaded properly and the application can fail. If the intent is to grant the permissions to all applications, use *file:${application}* as a code base in the grant entry.

Syntax of the policy file

A policy file contains several policy entries. The following example depicts each policy entry format:

```
grant [codebase <Codebase>] {
permission <Permission>;
 permission <Permission>;
permission <Permission>;
};

<CodeBase>:  A URL.
   For example,  "file:${java.home}/lib/tools.jar"
      When [codebase <Codebase>] is not specified, listed
               permissions are applied to everything.
```

```
      If URL ends with a JAR file name,  only the classes in the
               JAR file belong to the codebase.
               If URL ends with "/", only the class files in the specified
               directory belong to the codebase.
               If URL ends with "*", all JAR and class files in the specified
               directory belong to the codebase.
               If URL ends with "-", all JAR and class files in the specified
               directory and its subdirectories belong to the codebase.
<Permissions>: Consists from
       Permission Type   : class name of the permission
          Target Name       : name specifying the target
          Actions           : actions allowed on target
   For example,
         java.io.FilePermission "/tmp/xxx", "read,write"
```

Refer to developer kit specifications for the details of each permission.

Syntax of dynamic policy

You can define permissions for specific types of resources in dynamic policy files for an enterprise application. This action is achieved by using *product-reserved symbols*. The reserved symbol scope depends on where it is defined. If you define the permissions in the `app.policy` file, the symbol applies to all the resources on all of the enterprise applications that run on *node_name*. If you define the permissions in the `META-INF/was.policy` file, the symbol applies only to the specific enterprise application. Valid symbols for the code base are listed in the following table:

Table 78. Dynamic policy syntax.

This table describes valid symbols for the code base for dynamic policy files.

Symbol	Meaning
file:${application}	Permissions apply to all the resources within the application
file:${jars}	Permissions apply to all the utility Java archive (JAR) files within the application
file:${ejbComponent}	Permissions apply to the Enterprise JavaBeans (EJB) resources within the application
file:${webComponent}	Permissions apply to the web resources within the application
file:${connectorComponent}	Permissions apply to the connector resources within the application

You can specify the module name for a granular setting, except for these entries that are specified by the code base symbols. For example:

```
grant codeBase "file:DefaultWebApplication.war" {
   permission java.security.SecurityPermission "printIdentity";
 };

grant codeBase "file:IncCMP11.jar" {
permission java.io.FilePermission
"${user.install.root}${/}bin${/}DefaultDB${/}-",
"read,write,delete";
};
```

The sixth and seventh lines in the previous code sample are one continuous line. You can use a relative code base only in the `META-INF/was.policy` file. Several product-reserved symbols are defined to associate the permission lists to a specific type of resources.

Table 79. Dynamic policy syntax.

This table describes several product-reserved symbols that are defined to associate the permission lists to a specific type of resource.

Symbol	Meaning
file:${application}	Permissions apply to all the resources within the application
file:${jars}	Permissions apply to all the utility JAR files within the application
file:${ejbComponent}	Permissions apply to the enterprise beans resources within the application
file:${webComponent}	Permissions apply to the web resources within the application

Table 79. Dynamic policy syntax (continued).

This table describes several product-reserved symbols that are defined to associate the permission lists to a specific type of resource.

Symbol	Meaning
file:${connectorComponent}	Permissions apply to the connector resources both within the application and in the standalone connector resources.

Five embedded symbols are provided to specify the path and the name for the java.io.FilePermission permission. These symbols enable flexible permission specification. The absolute file path is fixed after the installation of the application.

Table 80. Dynamic policy syntax.

This table describes the embedded symbols that are provided to specify the path and name for the java.io.FilePermission permission.

Symbol	Meaning
${app.installed.path}	Path where the application is installed
${was.module.path}	Path where the module is installed
${current.cell.name}	Current cell name
${current.node.name}	Current node name
${current.server.name}	Current server name

Attention: Do not use the `${was.module.path}` in the `${application}` entry.

Carefully determine where to add a new permission. An incorrectly specified permission causes an AccessControlException exception. Because dynamic policy resolves the code base at runtime, determining which policy file has a problem is difficult. Add a permission only to the necessary resources. For example, `use ${ejbcomponent}`, and `etc` instead of `${application}`, and update the `was.policy` file instead of the `app.policy` file, if possible.

Static policy filtering

Limited static policy filtering support exists. If the `app.policy` file and the `was.policy` file have permissions that are defined in the `filter.policy` file with thefilterMask keyword, the runtime removes the permissions from the applications and an audit message is logged. However, if the permissions that are defined in the `app.policy` and the `was.policy` files are compound permissions, for example, java.security.AllPermission, the permission is not removed, but a warning message is written to the log file. The policy filtering only supports Developer Kit permissions; the permissions package name begins with `java` or `javax`.

Run-time policy filtering support is provided to force stricter filtering. If the `app.policy` file and the `was.policy` file have permissions that are defined in the `filter.policy` file with the runtimeFilterMask keyword, the runtime removes the permissions from the applications no matter what permissions are granted to the application. For example, even if a `was.policy` file has the java.security.AllPermission permission granted to one of its modules, specified permissions such as the runtimeFilterMask permission are removed from the granted permission during runtime.

Policy file editing

Using the policy tool that is provided by the Developer Kit (*app_server_root*`/java/jre/bin/policytool`), to edit the previous policy files is recommended. For WebSphere Application Server, Network Deployment, extract the policy files from the repository before editing. After the policy file is extracted, use the policy tool to edit the file. Check the modified policy files into the repository and synchronize them with other nodes.

Troubleshooting

To debug the dynamic policy, choose one of three ways to generate the detail report of the AccessControlException exception.

- **Trace** (Configured by RAS trace). Enables traces with the trace specification:

  Attention: The following command is one continuous line

```
com.ibm.ws.security.policy.*=all=enabled:
com.ibm.ws.security.core.SecurityManager=all=enabled
```

- **Trace** (Configured by property). Specifies a Java java.security.debug property. Valid values for the java.security.debug property are as follows:
  - Access. Print all debug information including required permission, code, stack, and code base location.
  - Stack. Print debug information including, required permission, code, and stack.
  - Failure. Print debug information including required permission and code.
- **ffdc**. Enable `ffdc`, modify the `ffdcRun.properties` file by changing `Level=4` and `LAE=true`. Look for an Access Violation keyword in the log file.

Access control exception for Java 2 security:

The Java 2 security behavior is specified by its *security policy*. The security policy is an access-control matrix that specifies which system resources certain code bases can access and who must sign them. The Java 2 security policy is declarative and it is enforced by the java.security.AccessController.checkPermission method.

The following example depicts the algorithm for the java.security.AccessController.checkPermission method. For the complete algorithm, refer to the Java 2 security check permission algorithm in the Security: Resources for learning article.

```
i = m;
while (i > 0) {
 if (caller i's domain does not have the permission)
  throw AccessControlException;
 else if (caller i is marked as privileged)
  return;
 i = i - 1;
};
```

The algorithm requires that all the classes or callers on the call stack have the permissions when a java.security.AccessController.checkPermission method is performed or the request is denied and a java.security.AccessControlException exception is created. However, if the caller is marked as *privileged* and the class (caller) is granted these permissions, the algorithm returns and does not traverse the entire call stack. Subsequent classes (callers) do not need the required permission granted.

A java.security.AccessControlException exception is created when certain classes on the call stack are missing the required permissions during a java.security.AccessController.checkPermission method. Two possible resolutions to the java.security.AccessControlException exception are as follows:

- If the application is calling a Java 2 security-protected application programming interface (API), grant the required permission to the application Java 2 security policy. If the application is not calling a Java 2 security-protected API directly, the required permission results from the side-effect of the third-party APIs accessing Java 2 security-protected resources.
- If the application is granted the required permission, it gains more access than it needs. In this case, it is likely that the third party code that accesses the Java 2 security-protected resource is not properly marked as privileged.

Example call stack

This example of a call stack indicates where application code is using a third-party API utility library to update the password. The following example is presented to illustrate the point. The decision of where to mark the code as privileged is application-specific and is unique in every situation. This decision requires great depth of domain knowledge and security expertise to make the correct judgement. A number of well written publications and books are available on this topic. Referencing these materials for more detailed information is recommended.

You can use the **PasswordUtil** utility to change the password of a user. The utility types in the old password and the new password twice to ensure that the correct password is entered. If the old password matches the one stored in the password file, the new password is stored and the password file updates. Assume that none of the stack frame is marked as privileged. According to the java.security.AccessController.checkPermission algorithm, the application fails unless all the classes on the call stack are granted write permission to the password file. The client application does not have permission to write to the password file directly and to update the password file at will.

However, if the PasswordUtil.updatePasswordFile method marks the code that accesses the password file as privileged, then the check permission algorithm does not check for the required permission from classes that call thePasswordUtil.updatePasswordFile method for the required permission as long as the **PasswordUtil** class is granted the permission. The client application can successfully update a password without granting the permission to write to the password file.

The ability to mark code privileged is very flexible and powerful. If this ability is used incorrectly, the overall security of the system can be compromised and security holes can be exposed. Use the ability to mark code privileged carefully.

Resolution to the java.security.AccessControlException exception

As described previously, you have two approaches to resolve a java.security.AccessControlException exception. Judge these exceptions individually to decide which of the following resolutions is best:
1. Grant the missing permission to the application.
2. Mark some code as privileged, after considering the issues and risks.

Enabling security for the realm

Use this topic to enable IBM WebSphere Application Server security. You must enable administrative security for all other security settings to function.

About this task

WebSphere Application Server uses cryptography to protect sensitive data and to ensure confidentiality and integrity of communications between WebSphere Application Server and other components in the network. Cryptography is also used by Web Services Security when certain security constraints are configured for the web services application.

Attention: Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

Note: Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

Complete the following steps to download and install the new policy files:

1. Click **Java SE 6**
2. Scroll down the page then click **IBM SDK Policy files**.

   The Unrestricted JCE Policy files for SDK 6 website displays.
3. Click **Sign in** and provide your IBM.com ID and password.
4. Select **Unrestricted JCE Policy files for SDK 6** and click **Continue**.
5. View the license and click **I Agree** to continue.
6. Click **Download Now**.
7. Extract the unlimited jurisdiction policy files that are packaged in the compressed file. The compressed file contains a `US_export_policy.jar` file and a `local_policy.jar` file.
8. In your WebSphere Application Server installation, go to the `$JAVA_HOME/jre/lib/security` directory and back up your `US_export_policy.jar` and `local_policy.jar` files.
9. Replace your `US_export_policy.jar` and `local_policy.jar` files with the two files that you downloaded from the IBM.com website.

Complete the following steps to enable security for the realm:

Procedure

1. Enable security in the WebSphere Application Server. Make sure that all node agents within the cell are active beforehand.

   For more information, see “Enabling security” on page 1172. It is important to click **Security > Global security**. Select an available realm definition from the list, and then click **Set as current** so that security is enabled upon a server restart.

   Note: In previous releases of WebSphere Application Server, the **Set as current** option is known as the **Enable global security** option.
2. Before restarting the server, log off the administrative console. You can log off by clicking **Logout** at the top menu bar.
3. Stop the server by going to the command line in the WebSphere Application Server *app_server_root*`/bin` directory and issue a `stopServer` *server_name* command.
4. Restart the server in secure mode by issuing the command `startServer` *server_name*. Once the server is secure, you cannot stop the server again without specifying an administrative user name and password. To stop the server once security is enabled, issue the command, `stopServer` *server_name* `-username` *user_id* `-password` *password*. Alternatively, you can edit the `soap.client.props` file in the *profile_root*`/properties` directory, and edit the `com.ibm.SOAP.loginUserid` or `com.ibm.SOAP.loginPassword` properties to contain these administrative IDs.

   If you have any problems restarting the server, review the output logs in the *profile_root*`/logs/server_name` directory. Check the Troubleshooting security configurations article for any common problems.

   The *app_server_root* variable refers to the *app_server_root*`/bin/` default directory.

Global security settings:

Use this panel to configure administration and the default application security policy. This security configuration applies to the security policy for all administrative functions and is used as a default security policy for user applications. Security domains can be defined to override and customize the security policies for user applications.

To view this administrative console page, click **Security > Global security**.

Security has some performance impacts on your applications. The performance impacts can vary depending upon the application workload characteristics. You must first determine that the needed level of security is enabled for your applications, and then measure the impact of security on the performance of your applications.

When security is configured, validate any changes to the user registry or authentication mechanism panels. Click **Apply** to validate the user registry settings. An attempt is made to authenticate the server ID or to validate the admin ID (if internalServerID is used) to the configured user registry. Validating the user registry settings after enabling administrative security can avoid problems when you restart the server for the first time.

Security configuration wizard:

Launches a wizard that enables you to configure the basic administrative and application security settings. This process restricts administrative tasks and applications to authorized users.

Using this wizard, you can configure application security, resource or Java 2 Connector (J2C) security, and a user registry. You can configure an existing registry and enable administrative, application, and resource security.

When you apply changes made by using the security configuration wizard, administrative security is turned on by default.

Security configuration report:

Launches a report that gathers and displays the current security settings of the application server. Information is gathered about core security settings, administrative users and groups, CORBA naming roles, and cookie protection. When multiple security domains are configured the report displays the security configuration associated with each domain.

A current limitation to the report is that it does not display application level security information. The report also does not display information on Java Message Service (JMS) security, bus security, or Web Services Security.

Enable administrative security:

Specifies whether to enable administrative security for this application server domain. Administrative security requires users to authenticate before obtaining administrative control of the application server.

For more information, see the related links for administrative roles and administrative authentication.

When enabling security, set the authentication mechanism configuration and specify a valid user ID and password (or a valid admin ID when internalServerID feature is used) in the selected registry configuration.

Note: There is a difference between the user ID (which is normally called the admin ID), which identifies administrators who manage the environment, and a server ID, which is used for server-to-server communication. You do not need to enter a server ID and password when you are using the internal server ID feature. However, optionally, you can specify a server ID and password. To specify the server ID and password, complete the following steps:

1. Click **Security > Global security**.
2. Under User accounts repository, select the repository and click **Configure**.
3. Specify the server ID and password in the Server user identity section.

Default: Enabled

Enable application security:

Enables security for the applications in your environment. This type of security provides application isolation and requirements for authenticating application users

In previous releases of WebSphere Application Server, when a user enabled global security, both administrative and application security were enabled. In WebSphere Application Server Version 6.1, the previous notion of global security is split into administrative security and application security, each of which you can enable separately.

As a result of this split, WebSphere Application Server clients must know whether application security is disabled at the target server. Administrative security is enabled, by default. Application security is disabled, by default. To enable application security, you must enable administrative security. Application security is in effect only when administrative security is enabled.

Default: Disabled

Use Java 2 security to restrict application access to local resources:

Specifies whether to enable or disable Java 2 security permission checking. By default, access to local resources is not restricted. You can choose to disable Java 2 security, even when application security is enabled.

When the **Use Java 2 security to restrict application access to local resources** option is enabled and if an application requires more Java 2 security permissions than are granted in the default policy, the application might fail to run properly until the required permissions are granted in either the `app.policy` file or the `was.policy` file of the application. AccessControl exceptions are generated by applications that do not have all the required permissions. See the related links for more information about Java 2 security.

Default: Disabled

Warn if applications are granted custom permissions:

Specifies that during application deployment and application start, the security runtime issues a warning if applications are granted any custom permissions. Custom permissions are permissions that are defined by the user applications, not Java API permissions. Java API permissions are permissions in the `java.*` and `javax.*` packages.

The application server provides support for policy file management. A number of policy files are available in this product, some of them are static and some of them are dynamic. Dynamic policy is a template of permissions for a particular type of resource. No code base is defined and no relative code base is used in the dynamic policy template. The real code base is dynamically created from the configuration and run-time data. The `filter.policy` file contains a list of permissions that you do not want an application to have according to the J2EE 1.4 specification. For more information on permissions, see the related link about Java 2 security policy files.

Important: You cannot enable this option without enabling the **Use Java 2 security to restrict application access to local resources** option.

Default: Disabled

Restrict access to resource authentication data:

Enable this option to restrict application access to sensitive Java Connector Architecture (JCA) mapping authentication data.

Consider enabling this option when both of the following conditions are true:

- Java 2 security is enforced.
- The application code is granted the accessRuntimeClasses WebSphereRuntimePermission permission in the `was.policy` file found within the application enterprise archive (EAR) file. For example, the application code is granted the permission when the following line is found in your `was.policy` file:

```
permission com.ibm.websphere.security.WebSphereRuntimePermission "accessRuntimeClasses";
```

The **Restrict access to resource authentication data** option adds fine-grained Java 2 security permission checking to the default principal mapping of the WSPrincipalMappingLoginModule implementation. You must grant explicit permission to Java 2 Platform, Enterprise Edition (J2EE) applications that use the WSPrincipalMappingLoginModule implementation directly in the Java Authentication and Authorization Service (JAAS) login when **Use Java 2 security to restrict application access to local resources** and the **Restrict access to resource authentication data** options are enabled.

Default: Disabled

Current realm definition:

Specifies the current setting for the active user repository.

This field is read-only.

Available realm definitions:

Specifies the available user account repositories.

The selections appear in a drop-down list containing:

- Local operating system
- Standalone LDAP registry
- Stand-alone custom registry

Configure...:

Select to configure the global security settings.

Web and SIP security:

Under Authentication, expand Web and SIP security to view links to:

- General settings
- Single sign-on (SSO)
- SPNEGO web authentication
- Trust association

General settings:

Select to specify the settings for web authentication.

Single sign-on (SSO):

Select to specify the configuration values for single sign-on (SSO).

With SSO support, web users can authenticate once when accessing both WebSphere Application Server resources, such as HTML, JavaServer Pages (JSP) files, servlets, enterprise beans, and Lotus Domino® resources.

SPNEGO web authentication:

Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) provides a way for web clients and the server to negotiate the web authentication protocol that is used to permit communications.

Trust association:

Select to specify the settings for the trust association. Trust association is used to connect reversed proxy servers to the application servers.

You can use the global security settings or customize the settings for a domain.

Note: The use of trust association interceptors (TAIs) for SPNEGO authentication is now deprecated. The SPNEGO web authentication panels now provide a much easier way to configure SPNEGO.

RMI/IIOP security:

Under Authentication, expand RMI/IIOP security to view links to:
- CSIv2 inbound communications
- CSIv2 outbound communications

CSIv2 inbound communications:

Select to specify authentication settings for requests that are received and transport settings for connections that are accepted by this server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

Authentication features include three layers of authentication that you can use simultaneously:
- **CSIv2 attribute layer**. The attribute layer might contain an identity token, which is an identity from an upstream server that already is authenticated. The identity layer has the highest priority, followed by the message layer, and then the transport layer. If a client sends all three, only the identity layer is used. The only way to use the SSL client certificate as the identity is if it is the only information that is presented during the request. The client picks up the interoperable object reference (IOR) from the namespace and reads the values from the tagged component to determine what the server needs for security.
- **CSIv2 transport layer**. The transport layer, which is the lowest layer, might contain a Secure Sockets Layer (SSL) client certificate as the identity.
- **CSIv2 message layer**. The message layer might contain a user ID and password or an authenticated token with an expiration.

CSIv2 outbound communications:

Select to specify authentication settings for requests that are sent and transport settings for connections that are initiated by the server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

Authentication features include three layers of authentication that you can use simultaneously:
- **CSIv2 attribute layer**. The attribute layer might contain an identity token, which is an identity from an upstream server that already is authenticated. The identity layer has the highest priority, followed by the message layer, and then the transport layer. If a client sends all three, only the identity layer is used.

The only way to use the SSL client certificate as the identity is if it is the only information that is presented during the request. The client picks up the interoperable object reference (IOR) from the namespace and reads the values from the tagged component to determine what the server needs for security.

- **CSIv2 transport layer**. The transport layer, which is the lowest layer, might contain a Secure Sockets Layer (SSL) client certificate as the identity.
- **CSIv2 message layer**. The message layer might contain a user ID and password or an authenticated token with an expiration.

Java authentication and authorization service:

Under Authentication, expand Java authentication and authorization service to view links to:

- Application logins
- System logins
- J2C authentication data

Application logins:

Select to define login configurations that are used by JAAS.

Do not remove the ClientContainer, DefaultPrincipalMapping, and WSLogin login configurations because other applications might use them. If these configurations are removed, other applications might fail.

System logins:

Select to define the JAAS login configurations that are used by system resources, including the authentication mechanism, principal mapping, and credential mapping.

J2C authentication data:

Select to specify the settings for the Java Authentication and Authorization Service (JAAS) Java 2 Connector (J2C) authentication data.

You can use the global security settings or customize the settings for a domain.

LTPA:

Select to encrypt authentication information so that the application server can send the data from one server to another in a secure manner.

The encryption of authentication information that is exchanged between servers involves the Lightweight Third-Party Authentication (LTPA) mechanism.

Kerberos and LTPA:

Select to encrypt authentication information so that the application server can send the data from one server to another in a secure manner.

The encryption of authentication information that is exchanged between servers involves the Kerberos mechanism.

Note: Kerberos must be configured before this option can be selected.

Kerberos configuration:

Select to encrypt authentication information so that the application server can send data from one server to anther in a secure manner.

The encryption of the authentication information that is exchanged between servers involves the KRB5 of LTPA mechanism.

Authentication cache settings:

Select to set your authentication cache settings.

Enable Java Authentication SPI (JASPI):

Select to enable the use of Java Authentication SPI (JASPI) authentication.

You can then click **Providers** to create or edit a JASPI authentication provider and associated authentication modules in the global security configuration.

Use realm-qualified user names:

Specifies that user names that are returned by methods, such as the getUserPrincipal() method, are qualified with the security realm in which they reside.

Security domains:

Use the Security Domain link to configure additional security configurations for user applications.

For example, if you want use a different user registry for a set of user applications than the one used at the global level, you can create a security configuration with that user registry and associate it with that set of applications. These additional security configurations can be associated with various scopes (cell, clusters/servers, SIBuses). Once the security configurations have been associated with a scope all of the user applications in that scope use this security configuration. Read about “Multiple security domains” on page 1222 for more detailed information.

For each security attribute, you can use the global security settings or customize settings for the domain.

External authorization providers:

Select to specify whether to use the default authorization configuration or an external authorization provider.

The external providers must be based on the Java Authorization Contract for Containers (JACC) specification to handle the Java(TM) 2 Platform, Enterprise Edition (J2EE) authorization. Do not modify any settings on the authorization provider panels unless you have configured an external security provider as a JACC authorization provider.

Custom properties:

Select to specify name-value pairs of data, where the name is a property key and the value is a string.

Specify extent of protection wizard settings:

Use this security wizard page to determine whether to enable application security and restrict access to local resources. When you use the wizard, admin security is enabled by default.

To view this security wizard page, click **Security > Global security > Security configuration wizard**.

Enable application security:

Enables security for the applications in your environment. This type of security provides application isolation and requirements for authenticating application users

In previous releases of WebSphere Application Server, when a user enabled global security, both administrative and application security were enabled. In WebSphere Application Server Version 6.1, the previous notion of global security is split into administrative security and application security, each of which you can enable separately.

As a result of this split, WebSphere Application Server clients must know whether application security is disabled at the target server. Administrative security is enabled, by default. Application security is disabled, by default. To enable application security, you must enable administrative security. Application security is in effect only when administrative security is enabled.

Default: Disabled

Use Java 2 security to restrict application access to local resources:

Specifies whether to enable or disable Java 2 security permission checking. By default, access to local resources is not restricted. You can choose to disable Java 2 security, even when application security is enabled.

When the **Use Java 2 security to restrict application access to local resources** option is enabled and if an application requires more Java 2 security permissions than are granted in the default policy, the application might fail to run properly until the required permissions are granted in either the `app.policy` file or the `was.policy` file of the application. AccessControl exceptions are generated by applications that do not have all the required permissions. See the related links for more information about Java 2 security.

Default: Disabled

Security custom properties:

Use this page to understand the psecurity.allowCustomHTTPMethodsredefined custom properties that are related to security.

To view this administrative console page, click **Security > Global security > Custom properties**. Then click **New** to add a new custom property and its associated value.

The custom properties in this topic are set in the administrative console through the previously listed path unless otherwise stated in the description.

You can use the custom properties page to define the following security custom properties:

- “com.ibm.audit.report.granularity” on page 1196
- “com.ibm.CSI.disablePropagationCallerList” on page 1196
- “com.ibm.CSI.propagateFirstCallerOnly” on page 1197
- “com.ibm.CSI.rmiInboundLoginConfig” on page 1197
- “com.ibm.CSI.rmiOutboundLoginConfig” on page 1197
- “com.ibm.CSI.supportedTargetRealms” on page 1197
- “com.ibm.security.multiDomain.setNamingReadUnprotected” on page 1198
- “com.ibm.security.useFIPS” on page 1198
- “com.ibm.websphere.crypto.config.certexp.notify.fromAddress” on page 1198
- “com.ibm.websphere.crypto.config.certexp.notify.textEncoding” on page 1198
- “com.ibm.websphere.lookupRegistryOnProcess” on page 1198

- “com.ibm.websphere.security.allowAnyLogoutExitPageHost” on page 1199
- “com.ibm.websphere.security.alwaysRestoreOriginalURL” on page 1199
- “com.ibm.websphere.security.config.inherit.trustedRealms” on page 1199
- “com.ibm.websphere.security.console.noSSLTreePortEndpoints” on page 1199
- “com.ibm.websphere.security.customLTPACookieName” on page 1199
- “com.ibm.websphere.security.customSSOCookieName” on page 1200
- “com.ibm.websphere.security.displayRealm” on page 1201
- “com.ibm.websphere.security.disableGetTokenFromMBean” on page 1201
- usec_seccustomprop.dita#com.ibm.websphere.security.enableAuditForIsCallerInRole
- “com.ibm.websphere.security.InvokeTAIbeforeSSO” on page 1202
- “com.ibm.websphere.security.JAASAuthData.addNodeNameSecDomain” on page 1202
- “com.ibm.websphere.security.JAASAuthData.removeNodeNameGlobal” on page 1202
- “com.ibm.websphere.security.krb.canonical_host” on page 1202
- “com.ibm.websphere.security.ldap.logicRealm” on page 1202
- “com.ibm.websphere.security.ldapSSLConnectionTimeout” on page 1203
- “com.ibm.websphere.security.logoutExitPageDomainList” on page 1203
- “com.ibm.websphere.security.performTAIForUnprotectedURI” on page 1203
-
- “com.ibm.websphere.security.rsaCertificateAliasCache” on page 1203
- “com.ibm.websphere.security.strictCredentialExpirationCheck” on page 1204
- “com.ibm.websphere.security.tokenFromMBeanSoapTimeout” on page 1204
- “com.ibm.websphere.security.useLoggedSecurityName” on page 1204
- “com.ibm.websphere.security.util.csiv2SessionCacheIdleTime” on page 1204
- “com.ibm.websphere.security.util.csiv2SessionCacheLimitEnabled” on page 1205
- “com.ibm.websphere.security.util.csiv2SessionCacheMaxSize” on page 1205
- “com.ibm.websphere.security.webAlwaysLogin” on page 1206
- “com.ibm.websphere.security.useLoggedSecurityName” on page 1204
- “com.ibm.ws.security.addHttpOnlyAttributeToCookies” on page 1206
- “com.ibm.ws.security.allowNonAdminToSecurityXML” on page 1207
- “com.ibm.ws.security.config.SupportORBConfig” on page 1207
- “com.ibm.ws.security.createTokenSubjectForAsynchLogin” on page 1207
- “com.ibm.ws.security.defaultLoginConfig” on page 1207
- “com.ibm.ws.security.failSSODuringCushion” on page 1207
- “com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA” on page 1208
- “com.ibm.ws.security.ssoInteropModeEnabled” on page 1208
- “com.ibm.ws.security.unprotectedUserRegistryMethods” on page 1208
- “com.ibm.ws.security.webChallengeIfCustomSubjectNotFound” on page 1209
- “com.ibm.ws.security.webInboundLoginConfig” on page 1209
- “com.ibm.ws.security.webInboundPropagationEnabled” on page 1209
-
-
- “com.ibm.wsspi.security.ltpa.tokenFactory” on page 1209
- “com.ibm.wsspi.security.token.authenticationTokenFactory” on page 1210
- “com.ibm.wsspi.security.token.authorizationTokenFactory” on page 1210
- “com.ibm.wsspi.security.token.propagationTokenFactory” on page 1210

- “com.ibm.wsspi.security.token.singleSignonTokenFactory” on page 1210
- “com.ibm.wsspi.wssecurity.kerberos.failAuthForExpiredKerberosToken” on page 1210
- “security.allowCustomHTTPMethods” on page 1210
- “security.enablePluggableAuthentication” on page 1211
- “security.useDefaultPolicyWhenJ2SDisabled” on page 1211

com.ibm.audit.report.granularity:

This property enables you to specify how much auditing data is recorded for each event type. If you only need to record basic information about an event, such as who did what action to what resource, and when, setting this property to `high`, might improve your application server performance.

You can specify values of `high`, `medium`, or `low` for this property. The default value is `low`.

Table 81. Type of data that is recorded for each event type based on the setting for com.ibm.audit.report.granularity. The following table indicates the type of data that is recorded for each event type based on the setting for this property.

Event type	`high` setting	`medium` setting	`low` setting
SessionContext	sessionId	sessionId, remoteHost	sessionId, remoteHost, remoteAddr, remotePort
PropagationContext (is only reported if SAP is enabled)	firstCaller (as part of the who)	firstCaller, and if verbose mode is enabled, the callerList	firstCaller, and if verbose mode is enabled, the callerList
RegistryContext	nothing is recorded	registry type	registry type
ProcessContext	nothing is recorded	realm	realm, and domain if verbose is enabled
EventContext	creationTime	creationTime, globalInstanceId	creationTime, globalInstanceId, eventTrailId, and lastTrailId if verbose mode is enabled
DelegationContext	identityName	delegationType, and identityName	delegationType, roleName, and identityName
AuthnContext	nothing is recorded	authn type	authn type
ProviderContext	nothing is recorded	provider	provider, and providerStatus
AuthnMappingContext	mappedUserName	mappedUserName, and mappedSecurityRealm	mappedUserName, mappedSecurityRealm, and mappedSecurityDomain
AuthnTermContext	terminateReason	terminateReason	terminateReason
AccessContext	progName, action, appUserName, and resourceName	progName, action, appUserName, resourceName, registryUserName, and accessDecision	progName, action, appUserName, resourceName, registryUserName, accessDecision, resourceType, permissionsChecked, permissionsGranted, rolesChecked, and rolesGranted
PolicyContext	nothing is recorded	policyName	policyName, and policyType
KeyContext	keyLabel	keyLabel, and keyLocation	keyLabel, keyLocation, and certificateLifetime
MgmtContext	nothing is recorded	mgmtType, and mgmtCommand	mgmtType, mgmtCommand, and targetInfoAttributes

com.ibm.CSI.disablePropagationCallerList:

This property completely disables the caller list and will not allow the caller list to change. This property prevents the creation of multiple sessions.

This property completely disables adding a caller or host list in the propagation token. Setting this property can be a benefit when the caller or host list in the propagation token is not needed in the environment.

Note: If this property is set to **true** as well as **com.ibm.CSI.propagateFirstCallerOnly**, then **com.ibm.CSI.disablePropagationCallerList** takes precedence.

Default false

com.ibm.CSI.propagateFirstCallerOnly:

This property will not allow the caller list to change and thus prevent the creation of multiple session entries. This property specifically limits the caller list to the first caller only.

This property logs the first caller in the propagation token that stays on the thread when security attribute propagation is enabled. Without setting this property, all caller switches get logged, which affects performance. Typically, only the first caller is of interest.

Note: If this property is set to **true** as well as **com.ibm.CSI.disablePropagationCallerList**, then **com.ibm.CSI.disablePropagationCallerList** takes precedence.

Default true

Note: New for this release, the default value of the com.ibm.CSI.propagateFirstCallerOnly security custom property is set to `true`. When this custom property is set to `true`, the first caller in the propagation token that stays on the thread is logged when security attribute propagation is enabled. When this property is set to `false`, all of the caller switches are logged, which can affect performance.

com.ibm.CSI.rmiInboundLoginConfig:

This property specifies the Java Authentication and Authorization Service (JAAS) login configuration that is used for Remote Method Invocation (RMI) requests that are received inbound.

By knowing the login configuration, you can plug in a custom login module that can handle specific cases for RMI logins.

Default system.RMI_INBOUND

com.ibm.CSI.rmiOutboundLoginConfig:

This property specifies the JAAS login configuration that is used for RMI requests that are sent outbound.

Primarily, this property prepares the propagated attributes in the Subject to be sent to the target server. However, you can plug in a custom login module to perform outbound mapping.

Default system.RMI_OUTBOUND

com.ibm.CSI.supportedTargetRealms:

This property enables credentials that are authenticated in the current realm to be sent to any realm that is specified in the Trusted target realms field. The Trusted target realms field is available on the CSIv2 outbound authentication panel. This property enables those realms to perform inbound mapping of the data from the current realm.

It is not recommended that you send authentication information to an unknown realm. Thus, this provides a way to specify that the alternate realms are trusted. To access the CSIv2 outbound authentication panel, complete the following steps:

1. Click **Security > Global security**.
2. Under RMI/IIOP security, click **CSIv2 outbound authentication**.

com.ibm.security.multiDomain.setNamingReadUnprotected:

This property can be set to true or false to determine if the CosNamingRead role protects all naming read operations. Setting this property to true is the equivalent of assigning the CosNamingRead role the Everyone special subject. If this propert is set, then it will override any assignments made to the CosNamingRead role.

Default none

com.ibm.security.useFIPS:

Specifies that Federal Information Processing Standard (FIPS) algorithms are used. The application server uses the IBMJCEFIPS cryptographic provider instead of the IBMJCE cryptographic provider.

Default false

com.ibm.websphere.crypto.config.certexp.notify.fromAddress:

This security property is used to customize the "from address" of certificate expiration notification email.

The value you assigned to this property should be an internet address, for example "Notification@abc-company.com" If this property is not set, WebSphere uses its email fromAddress: "WebSphereNotification@ibm.com" .

Default None

com.ibm.websphere.crypto.config.certexp.notify.textEncoding:

This security property is used to customize the text encoding character set for certificate expiration notification email.

WebSphere Application Server sends notification email for certificate expiration in either US-English or the machine default character set (if non-English locale is specified). If you want a different text encoding character set for the certificate expiration notification email, you can use this property to customize the text encoding character set.

Default None

com.ibm.websphere.lookupRegistryOnProcess:

This property can be set when realm registry lookups are performed via an MBean on a remote server if the realm is local OS security.

By default, the user registry tasks listRegistryUsers and listRegistryGroups perform lookups from the current process. In the case of Network Deployment (ND), that is the dmgr.

When dealing with a local OS user registry, lookup should occur on the actual server where the registry resides. In an ND environment that could be a remote machine. To perform lookup on the server process where the registry resides, the com.ibm.websphere.lookupRegistryOnProcess custom property should be set to `true`.

If com.ibm.websphere.lookupRegistryOnProcess is not set, or set to `false`, then the lookup is performed on the current process. The custom property can be set using the setAdminActiveSecuritySettings task for global security or the setAppActiveSecuritySettings task for a security domain.

com.ibm.websphere.security.allowAnyLogoutExitPageHost:

When you are using application form login and logout you can provide a URL for a custom logout page. By default, the URL must point to the host to which the request is made or to its domain. If this is not done, then a generic logout page is displayed rather than a the custom logout page. If you want to be able to point to any host, then you need to set this property in the `security.xml` file to a value of true. There is a risk that setting this property to have a value of true may open your systems to potential URL redirect attacks.

Default false

com.ibm.websphere.security.alwaysRestoreOriginalURL:

Use this property to indicate whether a cookie with the value WASReqURL is honored when the custom form login processor is used.

When this property is set to `true`, the value of WASReqURL takes precedence over the current URL, and the WASReqURL cookie is removed from subsequent requests.

When this property is set to `false`, the value of the current URL takes precedence, and the WASReqURL cookie is not removed from subsequent requests.

Default false

com.ibm.websphere.security.config.inherit.trustedRealms:

This property is used to inherit the global trusted realm settings from the global security configuration in the domain.

Security configuration trusted inbound and outbound realms are not inherited by default. However, there are some cases where the configuration might want to use (inherit) the settings from the global security configuration in the domain.

The value of this property can be either `true` or `false`.

com.ibm.websphere.security.console.noSSLTreePortEndpoints:

This property is used to improve the response time for large topology configurations.

When this property is set to `true` the status of the of the SSL port endpoints does not display on the Manage endpoint security configurations page in the administrative console. Displaying the status of the SSL port endpoints sometimes makes the administrative console seem like it is no longer functioning because of a longer than expected response time.

Default false

com.ibm.websphere.security.customLTPACookieName:

This property is used to customize the name of the cookies used for Lightweight Third Party Authentication (LTPA) tokens.

WebSphere Application Server Version 8.0 enables you to customize the name of the cookies used for LTPA and LTPA2 tokens. Custom cookie names allow you to logically separate authentication between Single Sign-On (SSO) domains and to enable customized authentication to a particular environment.

To take advantage of this functionality, a custom property must be set. For LTPA tokens, the custom property com.ibm.websphere.security.customLTPACookieName can be set to any valid string (special characters and spaces are not permitted) for the LTPA token cookie, and com.ibm.websphere.security.customSSOCookieName for the LTPA2 (SSO) token cookie. Each property is case-sensitive.

The value for this property is a valid string.

Note: Before you set this custom property, consider the following:

- This property, as with most custom properties, can be set at the security domain level. In this manner, a separate login can be forced between an administrative console login and an application login.
- The original default LTPAToken or LTPAToken2 cookie names are accepted and trusted by WebSphere Application Server Version 8.0. This enables compatibility with products such as Lotus Domino and WebSphere Portal which both utilize the default cookie name.
- Setting a custom cookie name can cause an authentication failure. For example, a connection to a server that has a custom cookie property set sends this custom cookie to the browser. A subsequent connection to a server that uses either the default cookie name or a different cookie name is not able to authenticate the request via a validation of the inbound cookie.
- This property does not function properly in a mixed-cell environment. For example, a deployment manager in WebSphere Application Server Version 8.0 might be able to create custom cookies. However, a WebSphere Application Server Version 7.0 node or server existing in this same cell does not understand what to do with this cookie and subsequently rejects it.
- If you utilize a product interacting with WebSphere Application Server that generates LTPA tokens, such as Lotus Domino or WebSphere Portal, be aware that these products might not be able to handle custom LTPA cookie names. Please consult the documentation for your product regarding its handling of custom LTPA cookie names.

Note: To activate this property, a restart of WebSphere Application Server is necessary.

com.ibm.websphere.security.customSSOCookieName:

This property is used to customize the name of the cookies used for Lightweight Third Party Authentication Version 2 (LTPA2) tokens.

WebSphere Application Server Version 8.0 enables you to customize the name of the cookies used for LTPA and LTPA2 tokens. Custom cookie names allow you to logically separate authentication between Single Sign-On (SSO) domains and to enable customized authentication to a particular environment.

To take advantage of this functionality, a custom property must be set. For LTPA tokens, the custom property com.ibm.websphere.security.customLTPACookieName can be set to any valid string (special characters and spaces are not permitted) for the LTPA token cookie, and com.ibm.websphere.security.customSSOCookieName for the LTPA2 (SSO) token cookie. Each property is case-sensitive.

The value for this property is a valid string.

Note: Before you set this custom property, consider the following:

- This property, as with most custom properties, can be set at the security domain level. In this manner, a separate login can be forced between an administrative console login and an application login.
- The original default LTPAToken or LTPAToken2 cookie names are accepted and trusted by WebSphere Application Server Version 8.0. This enables compatibility with products such as Lotus Domino and WebSphere Portal which both utilize the default cookie name.

- Setting a custom cookie name can cause an authentication failure. For example, a connection to a server that has a custom cookie property set sends this custom cookie to the browser. A subsequent connection to a server that uses either the default cookie name or a different cookie name is not able to authenticate the request via a validation of the inbound cookie.
- This property does not function properly in a mixed-cell environment. For example, a deployment manager in WebSphere Application Server Version 8.0 might be able to create custom cookies. However, a WebSphere Application Server Version 7.0 node or server existing in this same cell does not understand what to do with this cookie and subsequently rejects it.
- If you utilize a product interacting with WebSphere Application Server that generates LTPA tokens, such as Lotus Domino or WebSphere Portal, be aware that these products might not be able to handle custom LTPA cookie names. Please consult the documentation for your product regarding its handling of custom LTPA cookie names.

Note: To activate this property, a restart of WebSphere Application Server is necessary.

com.ibm.websphere.security.displayRealm:

This property specifies whether the HTTP basic authentication login window displays the realm name that is not defined in the application `web.xml` file.

Note: If the realm name is defined in the application `web.xml` file, this property is ignored.

If the realm name is not defined in the `web.xml` file, one of the following occurs:

- If the property is set to `false`, the WebSphere realm name display is Default Realm.
- If this property is set to `true`, the WebSphere realm name display is the user registry realm name for the LTPA authentication mechanism or the Kerberos realm name for the Kerberos authentication mechanism.

Important: If this property is set to `true`, and the user registry's realm name contains sensitive information, it is displayed to the user. For example, if standalone LDAP configuration is used, the LDAP server hostname and port are displayed. For LocalOS, the hostname is displayed.

Default	false
Type	string

com.ibm.websphere.security.disableGetTokenFromMBean:

Use this property to disables the outbound SOAP call to retrieve the subject from the originating server when Single Sign-On is enabled.

Typically, when Single Sign-On is enabled, and an inbound request needs to be authenticated, the receiving server attempts to retrieve the authentication from the originating server. The connection between the sending and receiving servers never times out during this callback process.

When this property is set to `true`, the receiving server does not attempt to authenticate the inbound request.

Default	false

com.ibm.websphere.security.enableAuditForIsCallerInRole:

Use this property to enable audit for the isCallerInRole method call.

If you set this property to `false`, it disables auditing for the invocation of isCallerInRole. In z/OS, SMF records are not issued for the invocation.

Default true

com.ibm.websphere.security.InvokeTAIbeforeSSO:

Default invocation order of Trust Association Interceptors (TAIs) in relation to Single Sign On (SSO) user authentication can be changed using this property. The default order is to invoke Trust Association Interceptors after SSO. This property is used to change the default order of TAI invocation with SSO. The property value is a comma (,) separated list of TAI class names to be invoked before SSO.

Default com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl
Type string

com.ibm.websphere.security.JAASAuthData.addNodeNameSecDomain:

By default, when JAAS authentication data entries are created at the domain security level, the alias name for the entry will be in the format *aliasName.* . You can enable the addition of the node name to the alias name in order to create the alias name in the format `nodeName/aliasName` for the entry, by setting the following property at the domain security level.

Note: You can set com.ibm.websphere.security.JAASAuthData.addNodeNameSecDomain=true at the global security level, to enable the addition of the node name to the alias name of JAAS authentication data entries for all security domains.

Default false

com.ibm.websphere.security.JAASAuthData.removeNodeNameGlobal:

By default, when JAAS authentication data entries are created at the global security level, the alias name for the entry will be in the format *nodeName/aliasName*. You can disable the addition of the node name to the alias name for the entry, by setting a value of `true` for this property at the global security level.

Default false

com.ibm.websphere.security.krb.canonical_host:

This custom property specifies whether the application server uses the canonical form of the URL/HTTP host name in authenticating a client. This property can be used for both SPNEGO TAI and SPNEGO Web.

If you set this custom property to `false`, a Kerberos ticket can contain a host name that differs from the HTTP host name header and the application server might issue the following message:

```
CWSPN0011E: An invalid SPNEGO token has been encountered while authenticating a HttpServletRequest
```

If you set this custom property to `true`, you can avoid this error message and allow the application server to authenticate using the canonical form of the URL/HTTP host name.

Default true

com.ibm.websphere.security.ldap.logicRealm:

This custom property enables you to change the name of the realm that is placed in the token.

This custom property enables you to configure each cell to have its own LDAP host for interoperability and backward compatibility. Also, it provides flexibility for adding or removing the LDAP host dynamically. If you are migrating a previous installation, this modified realm name does not take effect until administrative security is re-enabled. To be compatible with a previous release that does not support the logic realm, the name must be the same name that is used by the previous installation. You must use the LDAP host name, including a trailing colon and port number.

Type String

This property must be set as the custom property of a stand-alone LDAP registry. To set this custom property, in the administrative console:

1. Click **Security > Global security**.
2. Under User account repository, expand the Available realm definitions list, and select **Standalone LDAP registry**, and then click **Configure**.
3. Under Custom properties, click **New** , and then enter `com.ibm.websphere.security.ldap.logicRealm` in the **Name** field, and the new name of the realm that is placed in the token in the **Value** field.
4. Select this custom property and then click **Apply** or **OK**.

com.ibm.websphere.security.ldapSSLConnectionTimeout:

Use this property, when SSL is enabled on the LDAP server, to specify, in milliseconds, the maximum amount of time the JVM waits for a socket connection before issuing a timeout.

If one or more standalone LDAP servers are offline when a server process starts, and LDAP-SSL is enabled, there might be a delay of up to 3 minutes in the startup procedure even if you specify a value for the com.sun.jndi.ldap.connect.timeout custom property. When LDAP-SSL is enabled, any value specified for the com.sun.jndi.ldap.connect.timeout property is ignored.

When a value is specified for this property, the JVM tries to uses this connection timeout value when attempting to complete a socket connection, instead of trying to establish a directory context. When no value is specified for this property, the JVM tries to establish a directory context.

There will be no default value for this property.

com.ibm.websphere.security.logoutExitPageDomainList:

When you are using application form login and logout you can provide a URL for a custom logout page. By default, the URL must point to the host to which the request is made or to its domain. If this is not done, then a generic logout page is displayed rather than a the custom logout page. If you need to point to a different host, then you can populate this property in `security.xml` with a pipe (|) separated list of URLs that are allowed for the logout page.

Default none

com.ibm.websphere.security.performTAIForUnprotectedURI:

This property is used to specify TAI invocation behavior when **Use available authentication data when an unprotected URI is accessed** is selected in the administrative console.

Default false

com.ibm.websphere.security.rsaCertificateAliasCache:

This property is used to control the size of the alias cache.

The default value is 5000 and can be increased for larger deployments.

The value must be entered into the range of 1 - N, where `N` is a valid positive integer that is greater than or equal to the number of nodes registered with the Job Manager. You do not need to add this property unless your Job Manager topology exceeds 5000 registered nodes.

Default 5000

com.ibm.websphere.security.strictCredentialExpirationCheck:

Specifies whether credential expiration check occurs for a local EJB call. Typically, when an EJB invokes another EJB that is located in a local machine, a direct method invocation occurs even if the credentials of the original invoker expire before the local EJB call occurs.

If this property is set to `true`, a credential expiration check occurs on a local EJB call before the EJB is invoked on the local machine. If the credentials have expired, the EJB call is rejected.

If this property is set to false, a credential expiration check does not occur for a local EJB call.

Default false

com.ibm.websphere.security.tokenFromMBeanSoapTimeout:

Use this property to specify the amount of time the receiving server waits for an outbound SOAP call to retrieve the proper authentication from the originating server when Single Sign-On is enabled.

There is no default value for this property. If no value is specified, the global SOAP timeout value is used as the timeout value for the SOAP connection.

com.ibm.websphere.security.useLoggedSecurityName:

This is a custom property of user registries. This property alters the behavior of creating WSCredential.

A setting of `false` indicates that the security name returned by a user registry is always used to construct WSCredential.

A setting of `true` indicates that either a security name that is supplied by login module is used or a display name that was supplied by a user registry is used. This setting is compatible with WebSphere Application Server version 6.1 and older releases.

Default false

com.ibm.websphere.security.util.csiv2SessionCacheIdleTime:

This property specifies the time in milliseconds that a CSIv2 session can remain idle before being deleted. The session is deleted if the com.ibm.websphere.security.util.csiv2SessionCacheLimitEnabled custom property is set to a true value and the maximum size of the CSIv2 session cache is exceeded.

This custom property only applies if you enable stateful sessions, set the com.ibm.websphere.security.util.csiv2SessionCacheLimitEnabled custom property to true, and set a value for the com.ibm.websphere.security.util.csiv2SessionCacheMaxSize custom property. Consider decreasing the value for this custom property if your environment uses Kerberos authentication and has a short clock skew for the configured key distribution center (KDC). In this scenario, a short clock skew is defined as less than 20 minutes.

Important: Do not set a value for this function through the custom property panel because the value is not validated against the expected range of values. Instead, set the value on the CSIv2 outbound communications panel, which is available in the administrative console by completing the following steps:

1. Expand the **Security** section and click **Global security**.
2. Expand the **RMI/IIOP security** section and click **CSIv2 outbound communications**

You can set the value in the **Idle session timeout** field. However, when you specify this value on the CSIv2 outbound communications panel, the administrative console value is expected in seconds and not milliseconds.

The range of values for this custom property is 60,000 to 86,400,000 milliseconds. By default, the value is not set.

com.ibm.websphere.security.util.csiv2SessionCacheLimitEnabled:

This custom property specifies whether to limit the size of the CSIv2 session cache.

When you set this custom property value to `true`, you must set values for the com.ibm.websphere.security.util.csiv2SessionCacheIdleTime and com.ibm.websphere.security.util.csiv2SessionCacheMaxSize custom properties. When you set this custom property to `false`, the CSIv2 session cache is not limited. The default property value is false.

Consider setting this custom property to `true` if your environment uses Kerberos authentication and has a small clock skew for the configured key distribution center (KDC). In this scenario, a small clock skew is defined as less than 20 minutes. A small clock skew can result in a larger number of rejected CSIv2 sessions. However, with a smaller value for the com.ibm.websphere.security.util.csiv2SessionCacheIdleTime custom property, the application server can clean out these rejected sessions more frequently and potentially reduce the resource shortages.

Important: This custom property only applies if you enable the stateful sessions.

Important: Although you can enable the CSIv2 session cache limit option as a custom property, it is advisable that you enable the option on the CSIv2 outbound communications panel, which is available in the administrative console by completing the following steps:

1. Expand the **Security** section and click **Global security**.
2. Expand the **RMI/IIOP security** section and click **CSIv2 outbound communications**

You can enable the **Enable CSIv2 session cache limit** option. The default value is false.

com.ibm.websphere.security.util.csiv2SessionCacheMaxSize:

This property specifies the maximum size of the session cache after which expired sessions are deleted from the cache.

Expired sessions are defined as sessions that are idle longer than the time that is specified by the com.ibm.websphere.security.util.csiv2SessionCacheIdleTime custom property. When you use the com.ibm.websphere.security.util.csiv2SessionCacheMaxSize custom property, consider setting its value between 100 and 1000 entries.

Consider specifying a value for this custom property if your environment uses Kerberos authentication and has a small clock skew for the configured key distribution center (KDC). In this scenario, a small clock skew is defined as less than 20 minutes. Consider increasing the value of this custom property if the small cache size causes the garbage collection to run so frequently that it impacts the performance of the application server.

This custom property only applies if you enable stateful sessions, set the com.ibm.websphere.security.util.csiv2SessionCacheLimitEnabled custom property to true, and set a value for the com.ibm.websphere.security.util.csiv2SessionCacheIdleTime custom property.

Important: Do not set a value for this function through the custom property panel because the value is not validated against the expected range of values. Instead, set the value on the CSIv2 outbound communications panel, which is available in the administrative console by completing the following steps:

1. Expand the **Security** section and click **Global security**.
2. Expand the **RMI/IIOP security** section and click **CSIv2 outbound communications**

You can set the value in the **Maximum cache size** field.

The range of values for this custom property is 100 to 1000 entries. By default, the value is not set.

com.ibm.websphere.security.webAlwaysLogin:

This property specifies whether the login() method will throw an exception if an identity had already been authenticated. You can overwrite this behavior by setting this property to `true`.

Default	false
Type	string

Note: The login() method always uses the user ID and password to authenticate to the WebSphere application server irrespective of the presence of the SSO information in the HttpServletRequest.

com.ibm.ws.security.addHttpOnlyAttributeToCookies:

This custom property enables you to set the HTTPOnly attribute for single sign-on (SSO) cookies.

You can use the com.ibm.ws.security.addHttpOnlyAttributeToCookies custom property to protect cookies that contain sensitive values. When you set this custom property value to `true`, the application server sets the HTTPOnly attribute for SSO cookies whose values are set by the server. The HTTPOnly attribute enables the protection of sensitive values in cookies.

Also, a true value enables the application server to properly recognize, accept, and process inbound cookies with HTTPOnly attributes and inhibit any *cross-site scripting* from accessing sensitive cookie information.

A common security problem, which impacts web servers, is cross-site scripting. Cross-site scripting is a server-side vulnerability that is often created when user input is rendered as HTML. Cross-site scripting attacks can expose sensitive information about the users of the website. Most modern web browsers honor the HTTPOnly attribute to prevent this attack. A cookie with this attribute is called an *HTTPOnly cookie*. Information that exists in an HTTPOnly cookie is less likely to be disclosed to a hacker or a malicious website. For more information about the HTTPOnly attribute, see the Open Web Application Security Project (OWASP) website.

Important: When you use this custom property, HTTPOnly attribute is not added to every cookie that passes through the application server. Also, the attribute is not added to other non-secure cookies that are created by the application server. A list of non-HTTPOnly cookies includes:

- JSESSIONID cookies
- SSO cookies that are created by authenticators or providers from another software vendor
- Client or browser cookies that do not already contain the HTTPOnly attribute

You can set or remove this custom property from the Single sign-on panel in the administrative console by doing the following:

1. Click **Security > Global security**.
2. Under Authentication, click **Web and SIP security > Single sign-on (SSO)**.

Default true **Type** Boolean

com.ibm.ws.security.allowNonAdminToSecurityXML:

This property specifies whether the non-admin security roles are allowed the ability to modify the security.xml file. Setting this property to `true` gives non-admin security roles the ability to modify the security.xml file. In Version 6.1 and above, by default, non-admin security roles have the ability to modify the security.xml file.

Default false
Type Boolean

com.ibm.ws.security.config.SupportORBConfig:

Specifies whether to check or not check the object request broker (ORB) for properties. This property needs to be set as a system property. You set this property to `true` or `yes` so that the ORB is checked for properties. For any other setting, the ORB is completely ignored.

The property is to be used when a pluggable application client connects to the WebSphere Application Server. Specifically, this property is used whenever a hashmap containing security properties is passed in a hashmap on a new InitialContext(env) call.

com.ibm.ws.security.createTokenSubjectForAsynchLogin:

In this release, the actual LTPA token data is not available from a WSCredential.getCredentialToken() call when called from an asynchronous bean. For an existing configuration, you can add the `com.ibm.ws.security.createTokenSubjectForAsynchLogin` custom property and a `true` value to allow the LTPAToken to be forwarded to asynchronous beans. This property allows portlets to successfully perform LTPA token forwarding. This custom property is case sensitive. You must restart the application server after you add this custom property.

Note: This custom property applies only to system conditions where Server A makes EJB calls from asynchronous beans to Server B. This property does not apply for JAAS login situations.

Default not applicable

com.ibm.ws.security.defaultLoginConfig:

This property is the JAAS login configuration that is used for logins that do not fall under the WEB_INBOUND, RMI_OUTBOUND, or RMI_INBOUND login configuration categories.

Internal authentication and protocols that do not have specific JAAS plug points call the system login configuration that is referenced by com.ibm.ws.security.defaultLoginConfig configuration.

Default system.DEFAULT

com.ibm.ws.security.failSSODuringCushion:

Use the com.ibm.ws.security.failSSODuringCushion custom property to update custom JAAS Subject data for the LTPA token.

When you do not set this custom property to true, new JAAS Subjects might not contain the custom JAAS Subject data.

The default value is true.

com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA:

Use the com.ibm.ws.security.ltpa.forceSoftwareJCEProviderForLTPA custom property to correct an "invalid library name" error when you attempt to use a PKCS11 type keystore with a Java client.

The `ssl.client.props` file points to a configuration file, which in turn, points to the library name for the cryptographic device. The code for the Java client looks for a keystore type for the correct provider name. Without this custom property, the keystore type constant for PKCS11 is not specified correctly as it references the IBMPKCS11Impl provider instead. Also, the Lightweight Third Party Authentication (LTPA) code uses the provider list to determine the Java Cryptography Extension (JCE) provider. This approach causes a problem when Secure Sockets Layer (SSL) acceleration is attempted because the IBMPKCS11Impl provider needs to be listed before the IBMJCE provider within the `java.security` file.

This custom property corrects both issues so that SSL and other cryptographic mechanisms can use hardware acceleration.

Note: LTPA cannot use hardware acceleration because the software keys for LTPA do not implement the java.security.interfaces.RSAPrivateCrtKey interface, which is required by many accelerator cards.

Set this custom property to true when you want to use a PKCS11 type keystore with a Java client.

Default false

com.ibm.ws.security.ltpa.useCRT:

Use this property to improve the CPU utilization during the sign() operation that occurs when a new LTPA2 (SSO) token is created. When this property is set to `true`, the product implements the Chinese Remainder Theorem (CRT) algorithm when signing the new token. This property has no effect on the old style LTPA token.

Default false

com.ibm.ws.security.ssoInteropModeEnabled:

This property determines whether to send LtpaToken2 and LtpaToken cookies in the response to a web request (interoperable).

When this property value is false, the application server just sends the new LtpaToken2 cookie which is stronger, but not interoperable with some other products and Application Server releases prior to Version 5.1.1. In most cases, the old LtpaToken cookie is not needed and you can set this property to false.

Default true

com.ibm.ws.security.unprotectedUserRegistryMethods:

Specifies the method names on the UserRegistry interface, such as getRealm, getUsers, and isValidUser, that you do not want protected from remote access. If you specify multiple method names, separate the

names with either a space, a comma, a semi-colon, and a separator bar. See your implementation of the UserRegistry interface file for a complete list of valid method names.

If you specify an * as the value for this property, all methods are unprotected from remote access.

If a value is not specified for this property, all methods are protected from remote access.

If an attempt is made to remotely access a protected UserRegistry interface method, the remote process receives a CORBA NO_PERMISSION exception with minor code 49421098.

There is no default value for this property.

com.ibm.ws.security.webChallengeIfCustomSubjectNotFound:

This property determines the behavior of a single sign-on LtpaToken2 login.

If the token contains a custom cache key and the custom Subject cannot be found, then the token is used to log in directly as the custom information needs to be regathered if this property value is set to true. A challenge also occurs so that the user is required to login again. When this property value is set to false and the custom Subject is not found, the LtpaToken2 is used to login and gather all of the registry attributes. However, the token might not obtain any of the special attributes that downstream applications might expect.

Default true

com.ibm.ws.security.webInboundLoginConfig:

This property is the JAAS login configuration that is used for web requests that are received inbound.

By knowing the login configuration, you can plug in a custom login module that can handle specific cases for web logins.

Default system.WEB_INBOUND

com.ibm.ws.security.webInboundPropagationEnabled:

This property determines whether a received LtpaToken2 cookie should search for the propagated attributes locally before searching the original login server that is specified in the token. After the propagated attributes are received, the Subject is regenerated and the custom attributes are preserved.

Default true

com.ibm.wsspi.security.ltpa.tokenFactory:

This property specifies the Lightweight Third Party Authentication (LTPA) token factories that can be used to validate the LTPA tokens.

Validation occurs in the order in which the token factories are specified because LTPA tokens do not have object identifiers (OIDs) that specify the token type. The Application Server validates the tokens using each token factory until validation is successful. The order that is specified for this property is the most likely order of the received tokens. Specify multiple token factories by separating them with a pipe (|) without spaces before or following the pipe.

Default com.ibm.ws.security.ltpa.LTPATokenFactory |
com.ibm.ws.security.ltpa.LTPAToken2Factory |
com.ibm.ws.security.ltpa.AuthzPropTokenFactory

com.ibm.wsspi.security.token.authenticationTokenFactory:

This property specifies the implementation that is used for an authentication token in the attribute propagation framework. The property provides an old LTPA token implementation for use as the authentication token.

Default com.ibm.ws.security.ltpa.LTPATokenFactory

com.ibm.wsspi.security.token.authorizationTokenFactory:

This property specifies the implementation that is used for an authorization token. This token factory encodes the authorization information.

Default com.ibm.ws.security.ltpa.AuthzPropTokenFactory

com.ibm.wsspi.security.token.propagationTokenFactory:

This property specifies the implementation that is used for a propagation token. This token factory encodes the propagation token information.

The propagation token is on the thread of execution and is not associated with any specific user Subjects. The token follows the invocation downstream wherever the process leads.

Default com.ibm.ws.security.ltpa.AuthzPropTokenFactory

com.ibm.wsspi.security.token.singleSignonTokenFactory:

This property specifies the implementation that is used for a Single Sign-on (SSO) token. This implementation is the cookie that is set when propagation is enabled regardless of the state of the com.ibm.ws.security.ssoInteropModeEnabled property.

By default, this implementation is the LtpaToken2 cookie.

Default com.ibm.ws.security.ltpa.LTPAToken2Factory

com.ibm.wsspi.wssecurity.kerberos.failAuthForExpiredKerberosToken:

Use this property to specify how you want the system to handle authentication for a request after the Kerberos token for the request expires.

When this property is set to `true`, if a Kerberos token cannot be refreshed after it expires, authentication for the request fails.

When this property is set to `false`, authentication for the request does not fail even if the token has expired.

The default value for this property is false.

security.allowCustomHTTPMethods:

Use this custom property to permit custom HTTP methods. The custom HTTP methods are other than the standard HTTP methods, which are: DELETE, GET, HEAD, OPTIONS, POST, PUT or TRACE.

When this property is set to `false`, which is the default, if a combination of a URI pattern and a custom HTTP method are not listed in the security-constraint element, a search of the security constraint is performed using an URI pattern only. If there is a match, the value of the <auth-constraints> element is enforced. This behavior minimizes a potential security exposure.

When this property is set to `true`, the custom HTTP methods are treated as the standard HTTP methods. An authorization decision is made by both the URI pattern and the HTTP method. To properly protect a target URI, make sure that the proper HTTP methods are listed in the <web-resource-collection> element.

security.enablePluggableAuthentication:

This property is no longer used. Instead, use WEB_INBOUND login configuration.

Complete the following steps to modify the WEB_INBOUND login configuration:

1. Click **Security > Global security**.
2. Under Java Authentication and Authorization Service, click **System logins**.

Default true

security.useDefaultPolicyWhenJ2SDisabled:

The NullDynamicPolicy.getPermissions method provides an option to delegate a default policy class to construct a Permissions object when the `security.useDefaultPolicyWhenJ2SDisabled` custom property is set to `true`. When this property is set to `false`, an empty Permissions object is returned.

Default false

Security custom property collection:

Use this page to view and manage arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties.

The administrative console contains several custom properties pages that work similarly. To view one of these administrative pages, click a **Custom properties** link.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property is used.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in the application server.

Value:

Specifies the value paired with the specified name.

Description:

Provides information about the name-value pair.

Security custom property settings:

Use this page to configure arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

The administrative console contains several custom property settings pages that work similarly. To view one of these administrative pages, click **Custom properties**.

Name:

Specifies the name (or key) for the property.

Each property name must be unique. If the same name is used for multiple properties, the value specified for the first property is used.

Do not start your property names with `was.` because this prefix is reserved for properties that are predefined in the product.

Data type String

Value:

Specifies the value paired with the specified name.

Data type String

Description:

Provides information about the name and value pair.

Data type String

Testing security after enabling it

Basic tests are available that show whether the fundamental security components are working properly. Use this task to validate your security configuration.

Before you begin

After configuring administrative security and restarting all of your servers in a secure mode, validate that security is properly enabled.

There are a few techniques that you can use to test the various security login types. For example, you can test the Web-based BasicAuth login, Web-based form login, and the Java client BasicAuth login.

Basic tests are available that show whether the fundamental security components are working properly. Complete the following steps to validate your security configuration:

Procedure

1. After enabling security, verify that your system comes up in secure mode.
2. Test the Web-based BasicAuth with *Snoop*, by accessing the following URL: `http://hostname.domain:9080/snoop`.

3. Test the Web-based form login by starting the administrative console: `http://hostname.domain:port_number/ibm/console`. A form-based login page is displayed. If a login page does not appear, try accessing the administrative console by typing `https://myhost.domain:9043/ibm/console`.

   Type in the administrative user ID and password that are used for configuring your user registry when configuring security.
4. Test Java Client BasicAuth with *dumpNameSpace*.

   Use the `app_server_root/bin/dumpNameSpace` file. A login panel appears. If a login panel does not appear, there is a problem. Type in any valid user ID and password in your configured user registry.
5. Test all of your applications in secure mode.
6. If all the tests pass, proceed with more rigorous testing of your secured applications. If you have any problems, review the SYSOUT and SYSPRINT logs. For more information on common problems, see Troubleshooting security configurations.

Results

The results of these tests, if successful, indicate that security is fully enabled and working properly.

Security Configuration Wizard

The Security Configuration Wizard guides you through the process of completing the basic requirements to secure your application serving environment.

This wizard is available from the Security menu from the left pane of the admin console. To get to the wizard, navigate to **Security** > **Global security** > **Security Configuration Wizard**.

Step one of the configuration wizard allows you to choose the level of security desired. application-level security is selected by default. You also have the option of selecting Java 2 security.

Step two of the configuration wizard allows you to select a user repository. You have the following options:

- “Federated repository wizard settings” on page 1327
- “Local operating system wizard settings” on page 1260
- “Stand-alone custom registry wizard settings” on page 1291
- “Standalone LDAP registry wizard settings” on page 1267

Step three of the configuration wizard allows you to specify the local operating system user and group definitions as the repository, and, if necessary, to provide the name of a user with administrator privileges.

Step four of the configuration wizard provides a summary of the results of the configuration process.

Security configuration report

The security configuration report gathers and displays the current security settings of the application server. Information is gathered about core security settings, administrative users and groups, CORBA naming roles, and cookie protection. When multiple security domains are configured, each security domain has it's own report with a subset of the sections shown in the global security report that apply to the domain.

Note: The security configuration report now includes information about session security, web Attributes, and the HttpOnly setting to enable you to get a more complete view of your server security settings.

The report is a table with four columns: **Console Name**, **Security Configuration Name**, **Value** and **Console Path Name**. The security information gathered is divided into sections, and groups common security information. A row highlighted in blue with a title in the first column starts a new section.

The Security Configuration Report can be run from the administrative console by selecting **Security > Global Security** and then clicking **Security Configuration Report**. A new window displays the report information.

The columns

Console Name
Contains the name of the security attribute as found in the administrative console. If the value in this column is on a row highlighted in blue, and is the only entry on the row, then it is the start of a new section.

Security Configuration Name
Contains the security attribute as found in the configuration file.

Value Contains the value of the security attribute.

Console Path Name
Contains the path where the attribute is found on the console.

The sections

Security Settings
Displays information about the top-level security attributes. These attributes set the default for administrative security for the server, such as whether security is enabled, the default user registry, or if Java security is enabled.

For more information, read the Global security settings article.

Authentication Mechanisms and expirations
Contains all the attributes associated with each authentication mechanisms and trust associations as defined in the configuration.

User Registry
Displays the attributes for the default user registry for the server.

Authorization configuration
Displays attributes configured for an external Java Authorization Contract for Containers (JACC) provider.

Application login configuration
Displays application JAAS login entries and their login modules attributes.

For more information, read the SSL configurations collection article.

CSI Displays the attributes that define the inbound and outbound information for the Common Secure Interoperability (CSI) protocol.

SSL configuration repertoires
Displays the attributes that make up the Secure Sockets Layer (SSL) configuration used by the server. There can be multiple SSL configurations defined, and information about each is displayed. This object is often referenced by an SSL configuration group object used to associate it with an inbound or an outbound connection.

For more information, read the SSL configurations collection article.

Key stores
Displays the keystore attributes for each keystore in the configuration. Keystore objects in the configuration are often referenced by an SSL configuration object in the configuration.

For more information, read the Personal certificates collection article.

Trust managers
Displays the attributes that make up trust managers that can be used by the server. Trust manager objects in the configuration are typically referenced by an SSL configuration object.

For the more information, read the Trust managers collection article.

Key managers
Displays the attributes that make up the key managers that are used by the server. Key manager objects in the configuration are typically referenced by an SSL configuration object.

For more information, read the Key managers collection article.

SSL configuration group
Displays the attributes that make up an SSL configuration that are used for an outbound or an inbound connection.

Management scope
Displays the attributes that make up a management scope. The SSL configuration-related objects in the security configuration are defined within a management scope to reference the management scope object.

For more information, read the Management scope configurations article.

Key set groups
Displays the attributes that make up a group of key sets, which are used to manage public, private and shared keys.

For more information, read the Key set groups collection article.

Key set
Displays the attributes that make up the key set, which is used to manage public, private, and shared keys.

For more information, read the Key sets collection article.

Schedules
Displays the attributes that make up the scheduled process in the security configuration.

Notifications
Displays the attributes that make up notification objects in the security configuration.

Manage certificate expiration
Displays the attributes that define how startCertificateExpMonitor is run on the server.

System login configuration
Displays the attributes that define the System login entries and their login modules.

For more information, read the System login configuration entry settings for Java Authentication and Authorization Service article.

Custom properties
Displays all the custom properties that are defined in the security configuration.

For more information, read the Custom properties article.

Web Authentication
Displays properties that are used to define web authentication used by the server.

For more information, read the web authentication settings article.

Administrative Users and Groups
Displays the attributes that define roles and the users and groups associated with them as found in the `admin-authz.xml` file. The column titled **Administrative Role Name** contains the name of the administrative role. A column titled **Administrative Role Value** contains the user ID associated with the role (if one exists).

For more information, read the Administrative roles article.

Corba Naming Console Names
Displays the defined CORBA naming roles and the users that are assigned to the roles.

For more information, read the Administrative group roles and CORBA naming service groups article.

Console Name for Certificate Management
Lists all the certificate in keystore that are defined in the security configuration. There is also information about the certificates location and their validity period.

Cookie Protection
Displays attributes that pertain to HTTP Cookies. This section differs from other sections since information is gathered from different configuration files. The HttpOnly custom property, the web authentication com.ibm.wsspi.security.web.webAuthReq property, and the session security setting on each server are displayed on the report.

Java Authorization SPI Configuration
Displays the attributes that are defined for the Java Authorization SPI (JASPI) configuration. If there is a JASPI configuration object in the security configuration, information is included concerning whether JASPI is enabled, the name of the default JASPI provider, and a list of defined providers and their authentication modules.

Note: If JASPI has not been configured, this section is not shown in the security configuration report.

Adding a new custom property in a global security configuration or in a security domain configuration

Custom properties are arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure settings beyond those that are available in the administrative console. You can add new security custom properties in a security configuration or in a security domain configuration.

About this task

Adding a new custom property in a global security configuration using the administrative console

1. Click **Security > Global security > Custom properties**.
2. Click **New**,
3. Enter the property key name in the **Name** field.

   Each property key name must be unique. If the same name is used for multiple properties, the value specified for the first property is used.

   Do not start your property names with `was`, because this prefix is reserved for properties that are predefined in the application server.
4. Enter the property value in the **Value** field.
5. Click **Apply** or **Save**.

You can also use the -customProperties flag in the setAdminActiveSecuritySettings wsadmin command to add a new custom property in a global security configuration. See the SecurityConfigurationCommands command group for the AdminTask object article for more information about this command. For example:

```
wsadmin>AdminTask.setAdminActiveSecuritySettings('[-customProperties
["com.ibm.websphere.security.test=false"]]')
```

Adding a new custom property in a security domain configuration using the administrative console

1. Click **Security > Security domains**.
2. Select the global security domain you want to add a new custom property to.
3. Click **Custom properties**.
4. Click **New**.
5. Enter the property key name in the **Name** field.

Each property key name must be unique. If the same name is used for multiple properties, the value specified for the first property is used.

Do not start your property names with `was`, because this prefix is reserved for properties that are predefined in the application server.

6. Enter the property value in the **Value** field
7. Click **Apply** or **Save**.

You can also use the -customProperties flag in the setAppActiveSecuritySettings wsadmin command to add a new custom property in a global security domain configuration. See the SecurityConfigurationCommands command group for the AdminTask object article for more information about this command. Use the -securityDomainName flag to specify the security domain where the custom property is located. For example:

```
wsadmin>AdminTask.setAppActiveSecuritySettings('[ -securityDomainName testDomain
-customProperties ["com.ibm.websphere.security.test=false"]]')
```

Modifying an existing custom property in a global security configuration or in a security domain configuration

Custom properties are arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure settings beyond those that are available in the administrative console. You can modify existing security custom properties in a global security configuration or in a security domain configuration.

About this task

Modifying an existing custom property in a global security configuration using the administrative console

1. Click **Security > Global security > Custom properies**.
2. Select the custom property you want to modify.
3. Click **Edit** In the **Value** field, and then enter the value you want to modify.
4. Click **Apply** or **Save**.

You can also use the -customProperties flag in the setAdminActiveSecuritySettings wsadmin command to modify an existing custom property in a global security configuration. See the SecurityConfigurationCommands command group for the AdminTask object article for more information about this command. For example:

```
wsadmin>AdminTask.setAdminActiveSecuritySettings('[-customProperties
["com.ibm.websphere.security.test=false"]]')
```

Modifying an existing custom property in a security domain configuration using the administrative console

1. Click **Security > Security domains**.
2. Select the global security domain you want to modify.
3. Click **Custom properties**.
4. Select the custom property you want to modify.
5. Click **Edit**.In the **Value** field, and then enter the value you want to modify.
6. Click **Apply** or **Save**.

You can also use the -customProperties flag in the setAppActiveSecuritySettings wsadmin command to modify an existing custom property in a global security domain configuration. See the SecurityConfigurationCommands command group for the AdminTask object article for more information about this command. Use the -securityDomainName flag to specify the security domain where the custom property is located. For example:

```
wsadmin>AdminTask.setAppActiveSecuritySettings('[ -securityDomainName
testDomain -customProperties ["com.ibm.websphere.security.test=false"]]'
```

Deleting an existing custom property in a global security configuration or in a security domain configuration

Custom properties are arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure settings beyond those that are available in the administrative console. You can delete existing security custom properties in a global security configuration or in a security domain configuration.

About this task

Deleting an existing custom property in a global security configuration using the administrative console

1. Click **Security > Global security > Custom properties**.
2. Select the custom property you want to delete.
3. Click **Delete**.
4. Click **Apply** or **Save**.

You can also use the -customProperties flag in the setAdminActiveSecuritySettings wsadmin command to delete an existing custom property in a global security configuration. See the SecurityConfigurationCommands command group for the AdminTask object article for more information about this command. For example:

```
wsadmin>AdminTask.setAdminActiveSecuritySettings('[-customProperties
["com.ibm.websphere.security.test="]]')
```

Deleting an existing custom property in a security domain configuration using the administrative console

1. Click **Security > Security domains**.
2. Click **Custom properties**.
3. Select the custom property you want to delete.
4. Click **Delete**.
5. In the **Value** field, enter the value you want to delete.
6. Click **Apply** or **Save**.

You can also use the -customProperties flag in the setAppActiveSecuritySettings wsadmin command to delete an existing custom property in a global security domain configuration. See the SecurityConfigurationCommands command group for the AdminTask object article for more information about this command. Use the -securityDomainName flag to specify the security domain where the custom property is located. .For example:

```
wsadmin>AdminTask.setAppActiveSecuritySettings('[ -securityDomainName testDomain
-customProperties ["com.ibm.websphere.security.test="]]')
```

Configuring multiple security domains

By default, all administrative and user applications in WebSphere Application Server use the global security configuration. For example, a user registry defined in global security is used to authenticate users for every application in the cell. Out-of-the-box, this behavior is the same as it was in previous releases of WebSphere Application Server. You can create additional WebSphere security domains if you want to specify different security attributes for some or all of your user applications. This section describes how to configure a security domain by using the administrative console.

Before you begin

Only users assigned to the administrator role can configure or create new multiple security domains. Enable global security in your environment before configuring multiple security domains.

Read about “Multiple security domains” on page 1222 for a better understanding of what multiple security domains are and how they are supported in this version of WebSphere Application Server.

About this task

Security domains enable you to define multiple security configurations for use in your environment. For example, you can define different security (such as a different user registry) for user applications than for administrative applications. You can also define separate security configurations for user applications deployed to different servers and clusters.

Perform the following steps to configure a new security domain by using the administrative console:

Procedure

1. Click **Security > Security domains**.
2. If you are creating a new multiple security domain, click **New**. Supply a unique name and description for the domain and click **Apply**. If you want to configure an existing multiple security domain, select one to edit. Once you click **Apply** the domain name and additional sections are displayed. One section enables you to define the security attributes for the domain, and another section enables you to select the scopes to which the domain applies.
3. Under Assigned Scopes, select whether you want to assign the security domain to the entire cell or if you want to select the specific servers, clusters, and service integration buses to be included in the security domain. The Assigned Scopes section has two views. The default view is a cell topology. To assign the security domain to the entire cell, click the check box for the cell and then click **Apply** or **OK**.

   The name of the security domain appears next to the cell name, which indicates that the domain is now assigned to the cell. You can expand the topology and assign the domain to one or more servers and clusters. When an item in the topology is already assigned to another security domain, the check box is disabled and the name of the assigned domain is displayed to the right of the scope name. If you want to assign one of these scopes to the domain, you must first disassociate it with its current domain.

   Select **All assigned scopes** to view a list of only those resources that are currently assigned to the security domain.
4. Customize your security configuration by specifying security attributes for your new domain. Attributes that are not listed can not be customized at the domain level. Domains inherit attributes from the global security configuration.

   There are twelve individually configurable security attribute sections. You can expand and collapse each section. In the collapsed state, the name and a summary value for the section are displayed. Additionally, the summary value text indicates whether the attribute is defined in global security and is reused by the domain (as indicated by gray text) or if it is customized for the domain (as indicated by black text prefixed by the word “Customized”).

   Initially, each security attribute is set to use the global security settings. When an attribute is set to use global security, there is no domain-specific configuration for that attribute. Applications that use the domain use the global configuration for these security attributes.

   Only configure the security attributes that you want to change. To configure a security attribute for a domain, expand the security attribute section. The key properties of the global configuration display beneath the **Use global security** option. These properties are provided for convenience.

   To customize the configuration for the domain, select **Customize for this domain**. Configure the property and then click **OK** or **Apply**.

Note: In general, when you select **Customize for this domain**, you override all of the security configurations that are defined for that section in global security. Application logins, system logins, and J2C authentication data entries are some exceptions. When you define entries for a domain, applications in the domain are able to access the global entries in addition to the domain-specific entries.

For example, you might want to use a different user registry for applications that use the security domain but also want to use the global security configuration for all of the other security properties. In this case, expand the User Realm section and select **Customize for this domain**. Select a user registry type, click **Configure**, and provide the appropriate configuration details on the subsequent panel.

You can change security attributes such as the following:

Application Security

Specifies the settings for application security and Java 2 security. You can use the global security settings or customize the settings for a domain.

Select **Enable application security** to enable or disable security this choice for user applications. When this selection is disabled, all of the EJBs and web applications in the security domain are no longer protected. Access is granted to these resources without user authentication. When you enable this selection, the J2EE security is enforced for all of the EJBs and web applications in the security domain. The J2EE security is only enforced when Global Security is enabled in the global security configuration, (that is, you cannot enable application security without first enabling Global Security at the global level).

Java 2 Security

Select **Java 2 security** to enable or disable Java 2 security at the domain level. This choice enables or disables Java 2 security at the process (JVM) level so that all applications (both administrative and user) can enable or disable Java 2 security.

User realm

This section enables you to configure the user registry for the security domain. You can separately configure any registry that is used at the domain level. Read about “Multiple security domains” on page 1222 for more information.

Trust association

When you configure the trust association interceptor (TAI) at a domain level, the interceptors configured at the global level are copied to the domain level for convenience. You can modify the interceptor list at the domain level to fit your needs. Only configure those interceptors that are to be used at the domain level.

SPNEGO Web Authentication

The SPNEGO web authentication, which enables you to configure SPNEGO for web resource authentication, can be configured at the domain level.

Note: In WebSphere Application Server Version 6.1, a TAI that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. This function was deprecated in WebSphere Application Server 7.0. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

RMI/IIOP Security

The RMI/IIOP security attribute refers to the CSIv2 (Common Secure Interoperability version 2) protocol properties. When you configure these attributes at the domain level, the RMI/IIOP security configuration at the global level is copied for convenience.

You can change the attributes that need to be different at the domain level. The Transport layer settings for CSIv2 inbound communications should be the same for both the global and the domain levels. If they are different, the domain level attributes are applied to all of the application in the process.

JAAS application logins

Specifies the configuration settings for the Java Authentication and Authorization Service (JAAS) application logins. You can use the global security settings or customize the settings for a domain.

Note: The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. By default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

JAAS system logins

Specifies the configuration settings for the JAAS system logins. You can use the global security settings or customize the configuration settings for a domain.

JAAS J2C authentication

Specifies the configuration settings for the JAAS J2C authentication data. You can use the global security settings or customize the settings for a domain.

Java Authentication SPI (JASPI)

Specifies the configuration settings for a Java Authentication SPI (JASPI) authentication provider and associated authentication modules. You can use the global security settings or customize the settings for a domain. To configure JASPI authentication providers for a domain, select **Customize for this domain** and then enable JASPI. Select **Providers** to define providers for the domain.

Note: The JASPI authentication provider can be enabled with providers configured at the domain level. By default, all of the applications in the system have access to the JASPI authentication providers configured at the global level. The security runtime first checks for the JASPI authentication providers at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure JASPI authentication providers at a domain only when the provider is to be used exclusively by the applications in that security domain.

Authentication Mechanism Attributes

Specifies the various cache settings that need to applied at the domain level.

Select **Authentication cache settings** to specify your authentication cache settings. The configuration specified on this panel is applied only to this domain.

Select **LTPA Timeout** to configure a different LTPA timeout value at the domain level. The default timeout value is 120 minutes, which is set at the global level. If the LTPA timeout is set at the domain level, any token that is created in the security domain when accessing user applications is created with this expiration time.

When **Use realm-qualified user names** is enabled, user names returned by methods such as `getUserPrincipal( )` are qualified with the security realm (user registry) used by applications in the security domain.

Authorization Provider

You can configure an external third party JACC (Java Authorization Contract for Containers) provider at the domain level. Tivoli Access Manager's JACC provider can only be configured at

the global level. Security domains can still use it if they do not override the authorization provider with another JACC provider or with the built-in native authorization.

On z/OS, you can enable or disable the System Authorization Facility (SAF) based authorization at the domain level.

Custom properties

Set custom properties at the domain level that are either new or different from those at the global level. By default, all of the custom properties at the global security configuration can be accessed by all of the applications in the cell. The security runtime code first checks for the custom property at the domain level. If it does not find it, it then attempts to obtain the custom property from the global security configuration.

5. Once you have configured the security attributes and assigned the domain to one or more scopes, click **Apply** or **OK**.
6. Restart all servers and clusters for your changes to take effect.

Multiple security domains

The WebSphere Security Domains (WSD) provide the flexibility to use different security configurations in WebSphere Application Server. The WSD is also referred to as multiple security domains, or simply, security domains. You can configure different security attributes, such as the UserRegistry, for different applications.

Note: Multiple security domain support was introduced in WebSphere Application Server Version 7.0. You can create different security configurations and assign them to different applications in WebSphere Application Server processes. By creating multiple security domains, you can configure different security attributes for both administrative and user applications within a cell environment. You can configure different applications to use different security configurations by assigning the servers or clusters or service integration buses that host these applications to the security domains. Only users assigned to the administrator role can configure multiple security domains.

The following sections describe multiple security domains in more detail:

- “Why security domains are useful”
- “Relationship between global security and security domains” on page 1223
- “Contents of a security domain” on page 1224
- “Creating security domains” on page 1225
- “Configuring attributes for security domains” on page 1226
- “Associating scopes to security domains” on page 1227
- “Relationship between old server level security and the new security domains” on page 1228
- “How domain level security attributes are used by security runtime and applications” on page 1228
- “Client and application security programming model when using security domains” on page 1232
- “Application deployment in multiple domains configurations” on page 1233
- “Cross realm communication” on page 1234
- “Federating a node with security domains” on page 1236
- “Security domains in a mixed-version environment” on page 1237
- “Modifying security domains” on page 1237

Why security domains are useful

WebSphere Security Domains provide two major benefits:

- WebSphere Application Server administrative applications can be configured with a different set of security configurations from those for user applications.

- They enable one set of applications to have a different set of security configurations from another set of applications.

  For example, WebSphere Application Server administration can be configured to a user registry of federated repository while the applications can be configured to a user registry of LDAP.

In previous versions of WebSphere Application Server, all administrative and user applications use security attributes different from those attributes that are defined in global security. All administrative and user applications in WebSphere Application Server use global security attributes by default. For example, a user registry defined in global security is used to authenticate a user for every application in the cell.

In this release of WebSphere Application Server, however, you can use multiple security attributes for user applications other than the global security attributes, create a security domain for those security attributes that must differ, and associate them with the servers and clusters that host those user applications. You also can associate a security domain with the cell. All of the user applications in the cell use this security domain if they do not have a domain previously associated with them. However, global security attributes are still required for administrative applications such as the administrative console, naming resources and MBeans.

If you have used server level security in previous releases of WebSphere Application Server, you should now use multiple security domains since they are more flexible and easier to configure.

Server level security is deprecated in this release. Read “Relationship between global security and security domains” for more information.

Relationship between global security and security domains

Global Security applies to all administrative functions and the default security configuration for user applications. Security domains can be used to define a customized configuration for user applications.

You must have a global security configuration defined before you can create security domains. The global security configuration is used by all of the administrative applications such as the administrative console, naming resources, and Mbeans. If no security domains are configured, all of the applications use information from the global security configuration. User applications such as Enterprise JavaBeans (EJBs), servlets and administrative applications use the same security configuration.

When you create a security domain and associate it with a scope, only the user applications in that scope use the security attributes that are defined in the security domain. The administrative applications as well as the naming operations in that scope use the global security configuration. Define the security attributes at the domain level that need to be different from those at the global level. If the information is common, the security domain does not need to have the information duplicated in it. Any attributes that are missing in the domain are obtained from the global configuration. The global security configuration data is stored in the `security.xml` file, which is located in the `$WAS_HOME/profiles/$ProfileName/cells/$CellName` directory.

The following figure provides an example of a security multiple domain where the cell, a server and a cluster are associated with different security domains. As shown in the figure, the user applications in server `S1.1` as well as the cluster use security attributes that are defined in `Domain2` and `Domain3` respectively (since these scopes are associated with these domains). Server `S2.2` is not associated with a domain. As a result, the user application in `S2.2` uses the domain that is associated with the cell (`Domain1`) by default . Security attributes that are missing from the domain level are obtained from the global configuration.

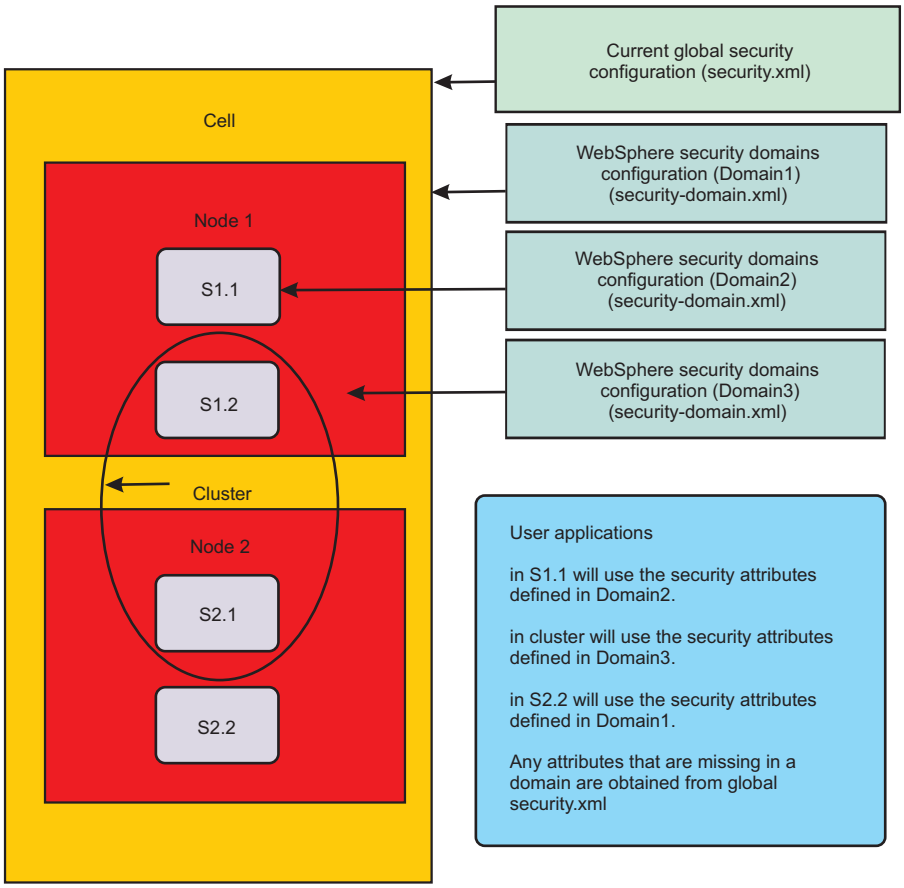


Figure 4. Scopes that can be associated to a security domain

The following figure shows the various high-level security attributes that can be defined at the global security configuration and those that can be overridden at the domain level.

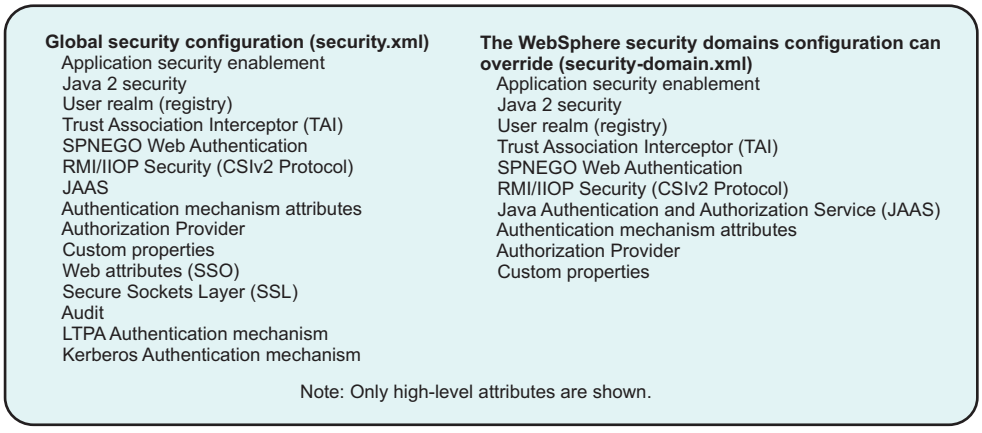


Figure 5. Security attributes that can be configured at the security domain

Contents of a security domain

A security domain is represented by two configuration files. One configuration file contains the list of attributes that are configured in the security domain. The other configuration file contains the scopes that use the security domain. The security domain information is stored in the `$WAS_HOME/profiles/`

$ProfileName/config/waspolicies/default/securitydomains/$SecurityDomainName directory. For every security domain that is configured, a $SecurityDomainName directory is created with two files in it: the security-domain.xml file contains the list of security attributes configured for the security domain, and the security-domain-map.xml file contains the scopes that use the security domain.

The following figure indicates the location of the main security domain related files and the contents of those files.

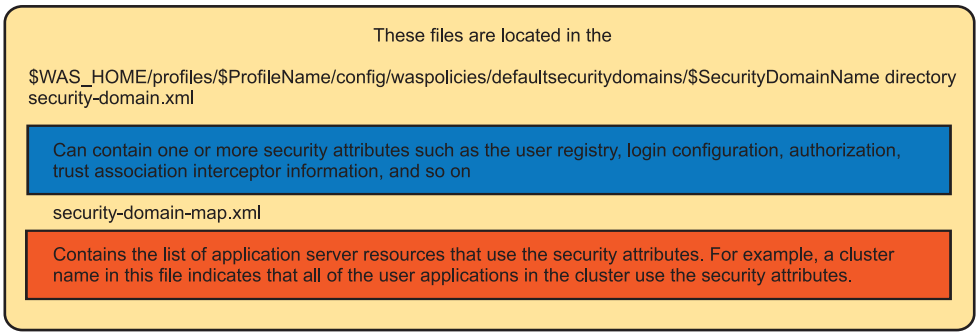


Figure 6. Location and contents of the main security domain related files

Note: You should not modify these files manually. Use administrative console tasks or scripting commands to modify the files instead. For a complete list of administrative tasks and scripting commands, see the links in "Related tasks" at the bottom of this document.

Creating security domains

Use the administrative console tasks or scripting commands to create security domains. In the administrative console, access security domains by clicking **Security > Security domains**. Help is available for each administrative console panel.

For a complete list of administrative console tasks and scripting commands, see the links in "Related tasks" at the bottom of this document.

When you create a security domain you must supply a unique name for the domain, the security attributes you want to configure for the security domain, and the scopes that need to use the security domain. Once configured, the servers that use the security domain must be restarted. The user applications in those scopes then use the attributes that are defined in the security domain. Any attributes that are not configured at the domain level are obtained from the global security configuration. Administrative applications and naming operations in the scopes always use the security attributes from the global security configuration. You must actively manage these attributes.

Any new security domain attributes must be compatible with those global security attributes that are inherited by the user applications that are assigned to the domain.

Other than for JAAS and custom properties, once global attributes are customized for a domain they are no longer used by user applications.

The security domains panel in the administrative console enables you to assign resources and to select the appropriate security attributes for your domain. The panel displays the key security attributes at the global configuration; you can make the decision to override them at the domain level if necessary. Once you have configured and saved the attributes at the domain level, the summary value on the panel displays the customized value for the domain (tagged with the word "customized" in black text).

A scope (a server, cluster, service integration bus or a cell) can be associated with only one domain. For example, you cannot define two domains that both have the cell-wide scope. Multiple scopes, however, can be defined in the same security domain. For example, a domain can be scoped to `Server1` and to `Server2` only within the cell.

The assigned scopes section on the security domain panel displays two views: one view that enables you to select and assign scopes to the domain, and another view that enables you to see a list of the currently assigned scopes. For convenience, you also have the flexibility to copy all of the security attributes from an existing security domain or the global configuration into a new security domain, and then modify only those attributes that must be different. You must still associate the scopes to these copied domains.

Scripting commands also provide you with the ability to create, copy and modify security domains. Once you create a domain, you must run the appropriate commands to associate security attributes and scopes to it.

Configuring attributes for security domains

Security attributes that can be configured at the domain level in WebSphere Application Server Version 8.0 are:

- Application security
- Java 2 security
- User realm (registry)
- Trust association
- Simple and Protected GSS-API Negotiation (SPNEGO) web authentication
- RMI/IIOP security (CSIv2)
- JAAS logins (Application, System and J2C Authentication Data)
- Java Authentication SPI
- Authentication mechanism attributes
- Authorization provider
- Federated repositories
- Custom properties

The security domains panels in the administrative console display all of these security attributes.

Some of the other well-known attributes that you cannot override at the domain level are Kerberos, Audit, Web Single Sign-on (SSO) and Tivoli Access Manager (TAM). The Secure Socket Layer (SSL) attribute already supports different scopes, but it is not part of the domain configuration. For all of the attributes that are not supported at the domain level, user applications in a domain share their configuration from the global level.

Any new security domain attributes must be compatible with those global security attributes that are inherited by the user applications that are assigned to the domain. You must actively manage these attributes. For example, if you customize only a JAAS configuration at the domain level you must make sure that it works with the user registry configured at the global level (if the user registry is not customized at the domain level).

Other than for JAAS and custom properties, once global attributes are customized for a domain they are no longer used by user applications.

The Tivoli Access Manager client runtime is used to provide authentication (used by TrustAssociationInterceptor and PDLoginModule) and authorization (used for JACC) by contacting TAM servers. There is only one Tivoli Access Manager runtime shared by all servers in a cell. Read the Tivoli Access Manager JACC provider configuration topic for more information.

You cannot have a different Tivoli Access Manager configuration at the security domain level to override the configuration at the cell level. However, you can to some degree specify Trust Association Interceptor (TAI) and JACC configuration at the security domain level. For example, you can use a different TAI or a different authorization provider. Since TAM server connectivity can only be defined at the global level, you can have a variety of TAIs defined and configured at the security domain level. Some of these TAIs might not use the TAM user repository, while others do. The TAIs that do need to connect to TAM will also connect to the globally-defined TAM server. Similarly, for authorization, you can have a variety of external authorization providers configured at the domain level. However, if any of these external authorization providers require connection to TAM they end up talking to the singular globally-configured TAM server.

Associating scopes to security domains

In WebSphere Application Server Version 8.0, you can associate a security domain at the cell level, the server level, the cluster level and the service integration bus level.

Note: For more information about the service integration bus and bus security in multiple security domains for WebSphere Application Server Version 8.0, see ../ae/cjr_sec_dom.dita.

When a security domain is associated with a server that is not part of a cluster, all user applications in that server use the attributes from the security domain. Any missing security attributes are obtained from the global security configuration. If the server is part of a cluster, you can associate the security domain with the cluster but not with the individual members in that cluster. The security behavior then remains consistent across all of the cluster members.

If a server is to be part of a cluster, create a cluster first and associate the security domain to it. You might have associated a domain to a server before it was a member of a cluster. If so, even though the domain is associated with the server directly, the security runtime code does not look at the domain. When a server is a cluster member, the security runtime disregards any security domains associated directly to the server. Remove the server scope from the security domain and associate the cluster scope to it instead.

A security domain can also be associated to the cell. This is usually done when you want to associate all user applications in WebSphere Application Server to a security domain. In this scenario, all of the administrative applications and the naming operations use the global security configuration while all of the user applications use the domain level configuration. If you want to split the security configuration information for administrative and user applications, this is all that is needed.

If you have a mixed-version environment, or plan to have one in future, and you want to associate security domains at the cell level, read “Security domains in a mixed-version environment” on page 1237 for more information.

If you are on a base profile server that has its own security domain defined, which is then federated to a deployment manager, associate the server scope to the security domain and not the cell scope. When you federate that node, the security domain information is propagated to the deployment manager. If the cell scope is associated to it, the network deployment configuration uses this security configuration, which might impact existing applications. During federation, the cell scope is changed to the server scope that is being federated. If the server scope is associated with the security domain, only that server uses the security domain after the federation. Other applications in other servers and clusters are not impacted. However, if this base profile server is registered to the Administrative Agent process you can associate the cell scope to the security domain if you want all of the servers from the base profile to use the same security domain for all of their user applications. Read about “Federating a node with security domains” on page 1236 for more information.

You can have a security domain associated at the cell level and also other security domains associated to various clusters or individual servers (those that are not part of any clusters). In this case, the security runtime first checks if any security domains are associated with the server or a cluster. If there is a security domain associated with the server or a cluster, the security attributes defined in it are used for all

of the applications in that server or cluster. Any security attributes missing from this server or cluster domain are obtained from the global security configuration, and not from the domain configuration associated with the cell.

If the server or cluster does not have its own domain defined, the security runtime code uses the security attributes from the domain associated with the cell (if one is defined). Any security attributes missing from the cell domain are inherited from the global security configuration.

Relationship between old server level security and the new security domains

In previous releases of WebSphere Application Server, you could associate a small set of security attributes at a server level. These attributes were used by all of the applications at the server level. The previous way of configuring the security attributes was deprecated in WebSphere Application Server 7.0, and will be removed in a future release.

You should now use the new security domains support starting in WebSphere Application Server 7.0, as these security domains are more easily managed and much more flexible. For example, in previous versions of WebSphere Application Server, you must manually associate the same security configuration to all of the cluster members by configuring the same security attributes for every server in a cluster.

The migration tool migrates the existing server level security configuration information to the new security domain configuration when the script compatibility mode is `false` (`-scriptCompatibility="false"`). A new security domain is created for every server security configuration if it is not part of a cluster. If it is part of a cluster, a security domain is associated with the cluster instead of with all of the servers in that cluster. In both cases, all of the security attributes that were configured at the server level in previous releases are migrated to the new security domain configuration, and the appropriate scope is assigned to the security domains.

If the script compatibility mode is set to `true`, the server level security configuration is not migrated to the new security domains configuration. The old server security configuration is migrated without any changes. The security runtime detects that the old security configuration exists and uses that information, even if a security domain is associated either directly or indirectly to the server. If the script compatibility mode is set to `true`, remove the security configuration from the server level and then create a security domain with the same set of security attributes.

How domain level security attributes are used by security runtime and applications

This section describes how the individual attributes at the domain level are used by the security runtime and how that impacts the user application security. Since all of these security attributes are also defined at the global level, more information about these attributes can be obtained elsewhere. For the purposes of this section, the emphasis is on domain level behavior.

1. **Application Security:**

   Select **Enable application security** to enable or disable security for user applications. When this selection is disabled, all of the EJBs and web applications in the security domain are no longer protected. Access is granted to these resources without user authentication. When you enable this selection, the J2EE security is enforced for all of the EJBs and web applications in the security domain. The J2EE security is only enforced when Global Security is enabled in the global security configuration, (that is, you cannot enable application security without first enabling Global Security at the global level).

2. **Java 2 Security:**

   Select **Use Java 2 security** to enable or disable Java 2 security at the domain level or to assign or add properties related to Java 2 security. This choice enables or disables Java 2 security at the process (JVM) level so that all applications (both administrative and user) can enable or disable Java 2 security.

3. **User Realm (User Registry):**

   This section enables you to configure the user registry for the security domain. You can separately configure any registry that is used at the domain level. Read about “Configuring attributes for security domains” on page 1226 for more information.

   When configuring a registry at the domain level you can choose to define your own realm name for the registry. The realm name distinguishes one user registry from another. The realm name is used in multiple places – in the Java client login panel to prompt the user, in the authentication cache, and when using native authorization.

   At the global configuration level, the system creates the realm for the user registry. In previous releases of WebSphere Application Server, only one user registry is configured in the system. When you have multiple security domains you can configure multiple registries in the system. For the realms to be unique in these domains, configure your own realm name for a security domain. You also can choose the system to create a unique realm name if it is certain to be unique. In the latter case, the realm name is based on the registry that is being used.

   For LDAP registries, the host:port of the LDAP server is the system-generated realm name. For localOS, the name of the localOS machine is the realm name. For custom user registries, the realm is the one returned by the `getRealm ( )` method of the custom registry implementation.

   If the system generated realm names are unique enough, you can choose the option for the system to generate the realm name. If not, choose a unique realm name for each security domain where you have the user registry configured. If the underlying user repository is the same, use the same realm name in different domains. From a security runtime perspective, same realm names have the same set of users and groups information. For example, when users and groups information is required from a realm, the first user repository that matches the realm is used.

   If a localOS registry that is not centralized is configured for any domain, and that domain is associated with servers or clusters in nodes not on the same system as the deployment manager, the realm name has to be provided. This realm name has to be the same as it would be if it were generated on the node. This realm name can be obtained by calling the `getRealm()` method on the SecurityAdmin MBean on that node. Typically, the realm name for localOS registries is the hostname of the machine. In this case, you should not let the system generate the realm name but rather get the realm name that is used by the processes in the node.

   If you select the system to generate the realm for the localOS registry at the time of the user registry configuration, it chooses the localOS registry that is used by the deployment manager. If the realm configured does not match the realm used by the servers then there are authorization issues. Also note that in this case, the domain using this local registry can only be associated with servers and clusters that belong to nodes on the same machine.

   Note: In WebSphere Application Server Version 7.0, the federated repositories user registry can only be configured at the global level and have only one instance per cell, but any domain can use it by configuring it as the active registry. In WebSphere Application Server Version 8.0, you can configure a unique instance of a federated repository at the domain level in a multiple security domain environment.

   When a security domain is copied from the global level, the users and groups defined at the global level are also copied to the security domain. This is also true when copying from an existing domain. A newly-created security domain that uses the file-based VMM repository requires that the user populate the repository with users and groups.

   Also new in this release of WebSphere Application Server, a new checkbox on the Realm configurations settings administrative console page, Use global schema for model, sets the global schema option for the data model in a multiple security domain environment. Global schema refers to the schema of the admin domain.

   When more than one user registry is in a process, the naming lookup that uses “UserRegistry” as the lookup name returns the user registry that is used by user applications. The user registry used by administrative applications is bound by the lookup name, “AdminUserRegistry”.

As described in “Cross realm communication” on page 1234, when an application in one realm communicates with an application in another realm using LTPA tokens, the realms have to be trusted. The trust relationship can be established using the **Trusted authentication realms – inbound** link in the user registry panel or by using the **addTrustedRealms** command. You can establish trust between different realms. A user logged into one realm can access resources in another realm. If no trust is established between the two realms the LTPA token validation fails.

Note: The realm name used in the web.xml file is not related to the user registry realm.

4. **Trust Association:**

   When you configure the trust association interceptor (TAI) at a domain level, the interceptors configured at the global level are copied to the domain level for convenience. You can modify the interceptor list at the domain level to fit your needs. Only configure those interceptors that are to be used at the domain level.

   Tivoli Access Manager's trust association interceptors can only be configured at the global level. The domain configuration can also use them, but cannot have a different version of the trust association interceptor. Only one instance of Tivoli Access Manager's trust association interceptors can exist in the cell.

5. **SPNEGO web authentication:**

   The SPNEGO web authentication, which enables you to configure SPNEGO for web resource authentication, can be configured at the domain level.

   Note: In WebSphere Application Server Version 6.1, a TAI that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function was deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

6. **RMI/IIOP Security (CSIv2):**

   The RMI/IIOP security attribute refers to the CSIv2 (Common Secure Interoperability version 2) protocol properties. When you configure these attributes at the domain level, the RMI/IIOP security configuration at the global level is copied for convenience.

   You can change the attributes that need to be different at the domain level. The Transport layer settings for CSIv2 inbound communications should be the same for both the global and the domain levels. If they are different, the domain level attributes are applied to all of the application in the process.

   When a process communicates with another process with a different realm, the LTPA authentication and the propagation tokens are not propagated to the downstream server unless that server is listed in the outbound trusted realms list. This can be done using the **Trusted authentication realms – outbound** link on the **CSIv2 outbound communication** panel, or by using the **addTrustedRealms** command task. Read about “Cross realm communication” on page 1234 for more information.

7. **JAAS (Java Authentication and Authorization Service):**

   The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. By default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

   For JAAS and custom properties only, once global attributes are customized for a domain they can still be used by user applications.

8. **Java Authentication SPI (JASPI)**

   Specifies the configuration settings for a Java Authentication SPI (JASPI) authentication provider and associated authentication modules to be applied at the domain level.

   Select **Providers** to create or to edit a JASPI authentication provider.

Note: The JASPI authentication provider can be enabled with providers configured at the domain level. By default, all of the applications in the system have access to the JASPI authentication providers configured at the global level. The security runtime first checks for the JASPI authentication providers at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure JASPI authentication providers at a domain only when the provider is to be used exclusively by the applications in that security domain.

9. **Authentication Mechanism Attributes:**

   Specifies the various cache settings that must be applied at the domain level.

   a. Authentication cache settings - use to specify your authentication cache settings. The configuration specified on this panel is applied only to this domain.

   b. LTPA Timeout - You can configure a different LTPA timeout value at the domain level. The default timeout value is 120 minutes, which is set at the global level. If the LTPA timeout is set at the domain level, any token that is created in the security domain when accessing user applications is created with this expiration time.

   c. Use realm-qualified user names - When this selection is enabled, user names returned by methods such as `getUserPrincipal( )` are qualified with the security realm (user registry) used by applications in the security domain.

10. **Authorization Provider:**

    You can configure an external third party JACC (Java Authorization Contract for Containers) provider at the domain level. Tivoli Access Manager's JACC provider can only be configured at the global level. Security domains can still use it if they do not override the authorization provider with another JACC provider.

    The JACC attributes, for example the Policy object, are based at the JVM level. This implies that there can be only be one JACC policy object in a JVM process. However, when you have multiple JACC providers configured, the deployment manager process has to handle all these providers in the same JVM because it has to propagate the authorization policy of applications to the respective provider based on the application name.

    If your JACC provider can handle propagating the authorization policy to multiple providers, you can configure it at the global level. In this case, when an application is installed, this JACC provider is called in the deployment manager process and it is the responsibility of this JACC provider to propagate the information to the corresponding JACC provider based on the application name passed in the contextID.

    Another way to achieve this is to set the custom property, `com.ibm.websphere.security.allowMultipleJaccProviders=true`, at the global security level. When this property is set, WebSphere Application Server propagates the authorization policy information to the JACC provider associated with the domain that corresponds to the target server where the application is installed. This property is only used at the deployment manager process since the managed servers do not host multiple JACC providers.

    On z/OS, you can enable or disable the System Authorization Facility (SAF) based authorization at the domain level.

11. **z/OS options:**

    You can set z/OS specific security options at the process (JVM) level so that all applications (both administrative and user) can enable or disable these options. These properties are:

    - Enabling application server and z/OS thread identity synchronization
    - Enabling the connection manager RunAs thread identity.

    For more information, read about z/OS security options.

12. **Custom properties:**

    Set custom properties at the domain level that are either new or different from those at the global level. By default, all of the custom properties at the global security configuration can be accessed by

all of the applications in the cell. The security runtime code first checks for the custom property at the domain level. If it does not find it, it then attempts to obtain the custom property from the global security configuration.

For JAAS and custom properties only, once global attributes are customized for a domain they can still be used by user applications.

Client and application security programming model when using security domains

A Java client or an application acting as a client that accesses an EJB typically does a naming lookup first. The naming resource, which is used by both administrative and the user applications, is considered an administrative resource. It is protected by the global security configuration information. In a multiple domain setup where the global security is using one realm (the *user registry*) and a domain is using a different realm, the Java client must authenticate to two different realms. The first authentication is required for the realm in the global security configuration for the naming operation to succeed, and the second authentication is required to access the EJB, which uses a different realm.

The `CosNamingRead` role protects all naming read operations. This role is usually assigned the **Everyone** special subject. This implies that any user, valid or not, can look up the name space. When a multiple domain is defined, if the `CosNamingRead` role has the **Everyone** special subject the security runtime code in the client side does not prompt you to log in. It uses the UNAUTHENTICATED subject to access the naming operation instead. Once the naming lookup operation is completed, when the client attempts to access the EJB it is prompted with a login panel that indicates the realm that is currently used by that EJB application (that is, the realm used in the domain). The client then presents the appropriate user credentials for that realm, which can then access the EJB. This logic applies to all variations of login source, including `properties` and `stdin`, not just when the login source is set to `prompt`.

If the **Everyone** special subject is removed from the `CosNamingRead` role, you are prompted twice. If the login source is `properties`, you can uncomment the com.ibm.CORBA.loginRealm property in the `$WAS_HOME/profiles/$ProfileName/properties/sas.client.props` file and add the appropriate realms using “|” as the separator. You must also enter the corresponding users and passwords in the com.ibm.CORBA.loginUserid and com.ibm.CORBA.loginPassword properties respectively. When you are using the programmatic logon in the Java client code you must authenticate twice with different user credentials; once prior to do a naming lookup for the EJB (the user should be in the global realm), and later prior to calling any method in the EJB (the user should be in the EJB domain's realm).

In general, when a Java client needs to authenticate to multiple and different realms it has to provide the credential information for all of those realms. If the login source is `prompt` or `stdin` it is prompted to login multiple times, once for each realm. If the login source is set to `properties`, the appropriate properties in the **sas.client.props** file (or any related file) are used for authenticating to different realms.

In certain scenarios, a client might make multiple calls to the same realm. For example, the Java client can access a resource using `realm1` followed by access to a resource using `realm2`, and then come back to access a resource in `realm1` again. In this case, the client is prompted three times; first for `realm1`, secondly for `realm2` and finally for `realm1` again.

By default, the subject that is used to login at a realm is not cached by the client side code. If you have this scenario, and you want the client to cache the subject based on the realm, set the com.ibm.CSI.isRealmSubjectLookupEnabled property to `true` in the **sas.client.props** file. If the com.ibm.CSI.isRealmSubjectLookupEnabled property is set, the client code caches the subject based on the realm name. The next time the Java client needs to authenticate to this realm, the cache is located to obtain the subject and the client is not prompted. Also, when the com.ibm.CSI.isRealmSubjectLookupEnabled property is set, the same subject that was logged in the first time is used for subsequent logins. If the subject information needs to change then this property should not be set.

If the client is doing a programmatic login it can pass the realm along with the user and password that it needs to authenticate. In this case, when the com.ibm.CORBA.validateBasicAuth property is set to `true` (the default value) in the **sas.client.props** file, the registry that matches the realm name is used for login. That realm must be supported in the process where the authentication takes place.

When using the WSLogin JAAS configurations, you also must set the `use_realm_callback` option in the **wsjaas_client.config** file in `$WAS_HOME/profiles/$ProfileName/properties` for the realm name to be passed to the call back handler. If you want to specify a different provider URL for the name server, set the `use_appcontext_callback` option and pass in the provider URL properties in a hash map to WSLogin.

If you do not know the realm name, use <*default*> as the realm name. The authentication is performed against the application realm. If the naming read operation does not have the **Everyone** special subject assigned, you must provide the realm that is used by the administrative applications (the registry used in the global security configuration), as well as the appropriate user and password information in that registry for the lookup operation to succeed.

After the lookup operation succeeds, perform another programmatic login by providing the application realm (or <*default*>) and the user and password information for the appropriate user in the registry that is used by the application. This is similar to the case where the login source is `prompt`. You must authenticate twice, once for the registry used by the global security configuration (for the naming lookup operation) and again for the registry used by the application to access the EJB.

If com.ibm.CORBA.validateBasicAuth is set to `false` in the `$WAS_HOME/profiles/$ProfileName/properties/sas.client.props` file then the programmatic login can use <*default*> as the realm name for both the lookup and the EJB operations. The actual authentication occurs only when the resource is accessed on the server side, in which case the realm is calculated based on the resource that is accessed.

The new security domain support starting in WebSphere Application Version 7.0 does not change the current application security programming model. However, it provides more flexibility and capabilities such as the following:

- User applications can still find the user registry object by using the naming lookup for “UserRegistry”. For the registry object used by administrative applications, the naming lookup for “AdminUserRegistry” can be used.
- The application usage of the JAAS login configuration does not change in a multiple domain setup. However, if an application must refer to the JAAS configuration that is specified at the domain level, the administrator and the deployer of that application must make sure that this domain is configured with the JAAS configurations that are required by the application.
- If an application needs to communicate with other applications using different realms, trust relationship should be established for both inbound and outbound communications when using the LTPA tokens. Read about “Cross realm communication” on page 1234 for more information.
- When using programmatic login in the applications, if you want to login to the realm used by the application, use <*default*> as the realm name or provide the realm name that the application is using. If you need to login to the global realm, you must provide the global realm name. If you provide any other realm, only a basic authentication subject is created. When the request actually flows to the server hosting that realm, the actual authentication of the user occurs if that server hosts the realm. If the server does not host the realm, the login fails.

Application deployment in multiple domains configurations

When deploying an application in a multiple domain setup, all of the modules in the application should be installed in the servers or clusters that belong to the same security domain. If not, depending on the security attributes configured in these security domains, inconsistent behavior can result. For example, if the domains contain different user registries, the users and groups information can be different, which can cause inconsistent behavior when accessing the modules. Another example is when the JAAS data is

different between the security domains. The JAAS configurations is not accessible from all of the modules in the application. The security runtime code and the command tasks rely on one domain being associated with an application when dealing with attributes such as user registry, JAAS login configurations, J2C authentication data, and authorization.

In most cases, application deployment fails when an application is deployed across different domains. However, since this was possible in earlier releases of WebSphere Application Server when only a few attributes were supported at the server level, the deployment tool first checks for attributes that are configured at the domains. If the attributes in the domain are the same as those supported in previous releases, the administrative console requests confirmation to ensure that you want to deploy application modules across multiple security domains. Unless there is an absolute requirement to deploy the applications across different domains, stop the deployment and select the servers and clusters in the same security domain.

Cross realm communication

When applications communicate using the RMI/IIOP protocol and LTPA is the authentication mechanism, the LTPA token is passed between the servers involved. The LTPA token contains the realm-qualified uniqueId, (also called the *accessId*), of the user who is logging into the front-end application. When this token is received by the downstream server it attempts to decrypt the token. If the LTPA keys are shared between the two servers, decryption succeeds and the accessId of the user is obtained from the token. The realm in the accessId is checked with the current realm that is used by the application. If the realms match, the LTPA token validation succeeds and it proceeds with the authorization. If the realms do not match, the token validation fails since the user from the foreign realm cannot be validated in the current realm of the application. If applications are not supposed to communicate with each other when using RMI/IIOP and the LTPA authentication mechanism, you do not to have to do anything further.

If you do want the cross realm communication to succeed when using RMI/IIOP and LTPA tokens, you must first establish trust between the realms involved, both for inbound and outbound communications.

For the server originating the request, its realm must have the realms that it can trust to send the token to. This is referred to as outboundTrustedRealms. For the server receiving the request, its realm needs to trust the realms that it can receive LTPA tokens from. This is referred to as inboundTrustedRealms.

Outbound trusted realms can be established using the addTrustedRealms command with the –communicationType option set to `outbound`. It can also be established in the administrative console by clicking **Trusted authentication realms - outbound** on the**CSIv2 outbound communications** panel.

Inbound trusted realms can be established using the same addTrustedRealms command task with the –communicationType option set to `inbound`. It can also be established by using the administrative console.

The figure below shows the communication between applications that use different user realms (registries) using RMI/IIOP. In this example, application `app1` (for example, a servlet) is configured to use the `realm1` user registry. The `app2` application (for example, an EJB) is configured to use the `realm2` user registry. The user (`user1`) initially logs into the servlet in `app1`, which then attempts to access an EJB in *app2*. The following must be set:

- In Domain1, `realm1` should trust `realm2` for the outbound communication.
- In Domain2, `realm2` should trust `realm1` for the inbound communication.
- The accessId for `user1` should be configured in the authorization table for `app2`.

When the LTPA token that contains the accessId of `user1` is received by `app2`, it decrypts the token. Both of the servers share the same LTPA keys. The LTPA token then ensures that the foreign realm is a trusted realm, and performs the authorization based on the accessId of `user1`. If security attribute propagation is not disabled, then the group information of `user1` is also propagated to `app2`. The groups can be used for

the authorization check, provided that the authorization table contains the group information. You can associate a special subject, AllAuthenticatedInTrustedRealms, to the roles instead of adding individual users and groups to the authorization table.

If the applications in the above example are deployed in different cells, you must do the following:

- Share the LTPA keys between the cells.
- Update the authorization table for `app2` with foreign users and groups accessIds by using the **wsadmin** utility. The administrative console does not have access to the realms outside of the scope of the cell.

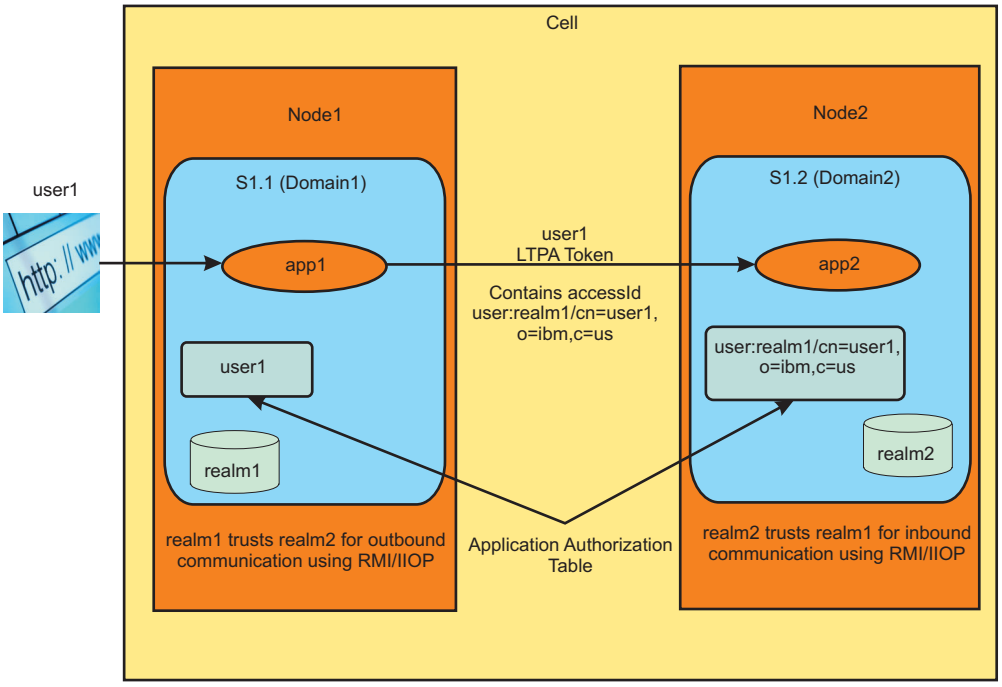


Figure 7. Cross realm communication in a multiple realm environment

Once trust has been established between the realms, when the server receives the LTPA token and the token is decrypted, it checks to see if the foreign realm is in its inbound trusted realms list. If it is trusted, the authentication succeeds. However, since it is a foreign realm, it does not go search the user registry to gather information about the user. Whatever information is in the LTPA token is used to authorize the user.

The only information in the LTPA token is the unique id of the user. This unique id of the user should exist in the authorization table for this application. If it does, authorization succeeds. However, if attribute propagation is enabled, additional authorization attributes (groups that this user belongs to) for the user are sent from the originating server to the receiving server. These additional attributes are used to make the access decisions. If the groups information exists in the propagation tokens it is used when making the authorization decision.

As previously mentioned, the information about the users and or the groups from the trusted realms should exist in the authorization table of the receiving application. Specifically, the accessId of the users and or groups should exist in the binding file of the application. This must be the case when the application is deployed. In the administrative console, when an application is deployed in a domain you can add the accessIds of the users and groups from any of its trusted realms to the authorization table.

You also have an option to associate a special subject, AllAuthenticatedInTrustedRealms, to the roles instead of adding individual users and groups. This is similar to the AllAuthenticated special subject that is currently supported. The difference is that the AllAuthenticated special subject refers to users in the same

realm as the application while the AllAuthenticatedInTrustedRealms special subject applies to all of the users in the trusted realms and in the realm of the application.

You can associate the accessId by using the $AdminApp install script. Because the accessId takes a unique format, use the command task listRegistryUsers with displayAccessIds set to `true`. If an invalid name or format is entered in this field, the authorization fails.

User and group information from the trusted realms is obtained by the deployment manager since it has access to all of the user registry configurations in all domains. However, in certain situations it is not possible to obtain the users and group information.

For example, if a server hosted on an external node is using localOS as the registry for its domain, the deployment manager cannot obtain the users and groups information unless it is running in the same operating system setup. The external operating system should be contacted to obtain this information. This can be done by directly invoking the registry in the server associated with that domain. The servers associated with the domain have to be started for this to work. You also must set the property, com.ibm.websphere.allowRegistryLookupOnProcess, to `true` in the top-level security custom properties. When this property is set, the deployment manager code searches one of the servers that is associated with the security domain and obtains the users and groups information directly from it. This is possible by calling an MBean in one of the servers.

If the MBean in any of the servers that are using that domain cannot be accessed, the administrative console displays a panel where you can enter the user and accessId information manually for each user and group. It is important that the correct accessId format be entered in this field. The accessId format for the user is `user:realmName/userUniqueId`. The realmName is the name of the realm where the user resides, and the userUniqueId is the uniqueId that represents the user, depending on the registry that is used.

For example, for LDAP, the uniqueUserId is the Distinguished Name (DN), for the Windows localOS registry and is the SID of the user. For Unix platforms, it is the UID. For custom registries, it depends on the implementation.

Similarly, for groups, the accessId format is group:realmName/groupUniqueId. As previously mentioned, use the listRegistryUsers and listRegistryGroups command with the –displayAccessIds option set to `true` so that you can obtain the correct format for the domain or realm that you are interested in.

Once users and groups from the trusted realms or the AllAuthenticatedInTrustedRealms special subject is added to the authorization table of the application, it is ready to accept requests from other applications that are using any of its trusted realms. The LTPA token validation on the receiving server first checks to make sure that the realm is trusted. The authorization engine then checks to see if the external user and/or the groups or the AllAuthenticatedInTrustedRealms special subject are given access to the roles needed to access the resource. If true, access is granted.

Cross realm communication is only applicable when using the WebSphere built-in authorization. If you are using other authorization engines including SAF for z/OS, any cross realm authorization can be achieved by implementing custom login modules that map external users to users in its own repository.

Federating a node with security domains

When a security domain is configured in the base version and is federated to a cell, the security domain configured at the base version is also configured for that server in the cell. The same domain security configuration can be used by the server before and after the federation. If a base server is to be federated to a cell, the resource assigned to the security domain should be the server scope instead of the cell scope.

If the base server is expected to be registered with an Administrative Agent process, use the cell scope as the resource if the intention is to have all of the servers in the base profile use this security domain.

If during federation the security domain at the base already exists at the cell level, the **addNode** command fails. You can use the –excludesecuritydomains option not to include the security domain during federation.

When the federated node is removed from a cell, the resources in that node should be removed from the security domains. If security domains have clusters associated with them that span nodes, the nodes are not removed. You can always remove resources from the security domains or any domains that are not used by using scripting commands or the administrative console.

Security domains in a mixed-version environment

You should create security domains once all of the nodes have been migrated to the latest version. This is especially true if there is a need to associate the cell with a domain. However, if you want to create security domains in a mixed- version environment, be aware of the following:

- If a cell-wide domain is created in a mixed version setup, a domain called *PassThroughToGlobalSecurity* is created automatically. All mixed clusters are assigned to this domain at the time of the creation of the cell-wide domain. This PassThroughToGlobalSecurity domain is special in the sense that attributes cannot be added to it, only resources can be assigned to it.

  All resources assigned to the PassThroughToGlobalSecurity domain use the global security configuration information. Whenever a node in the mixed version setup is migrated to the latest version, the servers and clusters in these nodes are added to this domain. Applications in all of the servers and clusters in these nodes do not use the cell-wide domain; they instead use the global security configuration before and after migration.

  If any of these servers need to use the cell-wide domain, you must remove these resources from this PassThroughToGlobalSecurity domain. New servers and clusters that are created in the migrated node use the cell-wide domain, not the PassThroughToGlobalSecurity domain. As a result, you have a mix of servers and clusters, some of them using global security configuration and some using the cell-wide domain.
- Once a cell-wide domain is created, adding any old version cluster members to a WebSphere Application Server Version 8.0 cluster is restricted since this action makes it a mixed cluster. This restriction also holds true when a WebSphere Application Server Version 8.0 cluster is associated with a domain. and a previous version cluster member is added to this cluster. This restriction is needed to avoid associating a security domain to a mixed cluster.
- If possible, you should create a cell-wide domain after all of the nodes have been migrated. In this case, the cell-wide domain is applicable to the entire cell and not just to parts of it. This also eliminates the need to create the PassThroughToGlobalSecurity domain and the mixed cluster scenario with security domains.

Modifying security domains

Use the administrative console tasks or scripting commands to modify security domains. For a complete list of administrative tasks and scripting commands, see the links in "Related tasks" at the bottom of this document.

Once a security domain is created and associated to a set of scopes, the servers associated with this new domain must be restarted. After the restart, the applications in the scopes associated with the new domain use the security attributes defined in the domain.

Changes to any of the domain attributes requires the restart of all of the scopes assigned to it. If new scopes are added they also need to be restarted. Any modifications to the domain configuration, either to the security attributes or to the scopes, has impacts on those applications that are using the domain configuration.

Before you make modifications to an existing domain, consider the following potential impacts. For example, if a user registry that is configured at a domain is removed, and the servers restarted, the user registry from the cell-wide domain (if one is defined), or the global security configuration is then used. This can impact application authentication and authorization. Users and groups associated with an application might no longer be valid in the new registry. Another example to consider is when JAAS configurations are removed from a domain. Applications that rely on this are no longer be able to use the JAAS configurations. Whenever a security configuration is changed it might impact your applications, so all security configuration changes should be made with the utmost care.

Creating new multiple security domains

You can create multiple security domains in your configuration. By creating multiple security domains, you can configure different security attributes for administrative and user applications within a cell environment.

Before you begin

Only users assigned to the administrator role can create new multiple security domains. Enable global security in your environment before creating new multiple security domains.

Read about "Multiple security domains" on page 1222 for a better understanding of what multiple security domains are and how they are supported in this version of WebSphere Application Server.

About this task

Security domains provide a mechanism to use different security settings for administrative applications and user applications. They also provide the ability to support multiple security settings so different applications can use different security attributes like user registry or login configurations.

Use multiple security domains to achieve the following goals:

- Configure different security attributes for administrative and user applications within a cell
- Consolidate server configurations by managing different security configurations within a cell
- Restrict access between applications with different user registries, or configure trust relationships between applications to support communication across registries

Perform the following steps to create a new security domain using the administrative console:

Procedure

1. Click **Security > Security domains**.
2. On the Security domains collection page, click **New**.
3. Specify a unique name for the domain. A domain name must be unique within a cell and cannot contain an invalid character. This field is required.
4. Specify a unique description for the domain. After you click **Apply** you are returned to the Security domains detail page
5. Under Assigned Scopes, assign the security domain to the entire cell or select the specific servers, clusters, and service integration buses to include in the security domain.
6. Customize your security configuration by specifying security attributes for your new domain and by assigning it to cell resources.

   You can change security attributes such as the following:

   Application Security

   Specifies the settings for application security and Java 2 security. You can use the global security settings or customize the settings for a domain.

   Select **Enable application security** to enable or disable security this choice for user applications. When this selection is disabled, all of the EJBs and web applications in the

security domain are no longer protected. Access is granted to these resources without user authentication. When you enable this selection, the J2EE security is enforced for all of the EJBs and web applications in the security domain. The J2EE security is only enforced when Global Security is enabled in the global security configuration, (that is, you cannot enable application security without first enabling Global Security at the global level).

Java 2 Security

Select **Java 2 security** to enable or disable Java 2 security at the domain level. This choice enables or disables Java 2 security at the process (JVM) level so that all applications (both administrative and user) can enable or disable Java 2 security.

User realm

This section enables you to configure the user registry for the security domain. You can separately configure any registry that is used at the domain level. Read about “Multiple security domains” on page 1222 for more information.

Trust association

When you configure the trust association interceptor (TAI) at a domain level, the interceptors configured at the global level are copied to the domain level for convenience. You can modify the interceptor list at the domain level to fit your needs. Only configure those interceptors that are to be used at the domain level.

SPNEGO Web Authentication

The SPNEGO web authentication, which enables you to configure SPNEGO for web resource authentication, can be configured at the domain level.

Note: In WebSphere Application Server Version 6.1, a TAI that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. This function was deprecated in WebSphere Application Server Version 7.0. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

RMI/IIOP Security

The RMI/IIOP security attribute refers to the CSIv2 (Common Secure Interoperability version 2) protocol properties. When you configure these attributes at the domain level, the RMI/IIOP security configuration at the global level is copied for convenience.

You can change the attributes that need to be different at the domain level. The Transport layer settings for CSIv2 inbound communications should be the same for both the global and the domain levels. If they are different, the domain level attributes are applied to all of the application in the process.

JAAS application logins

Specifies the configuration settings for the Java Authentication and Authorization Service (JAAS) application logins. You can use the global security settings or customize the settings for a domain.

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. Be default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

JAAS system logins

Specifies the configuration settings for the JAAS system logins. You can use the global security settings or customize the configuration settings for a domain.

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. Be default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

Note: For both JAAS application logins and JAAS system logins, the collections are not populated until one is created first. You can do this by selecting **customize for this domain** under JAAS application logins or JAAS system logins and then by selecting **Apply** or **OK**.

JAAS J2C authentication

Specifies the configuration settings for the JAAS J2C authentication data. You can use the global security settings or customize the settings for a domain.

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. Be default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

Java Authentication SPI (JASPI)

Specifies the configuration settings for a Java Authentication SPI (JASPI) authentication provider. You can use the global security settings or customize the settings for a domain. To configure JASPI authentication providers for a domain, select **Customize for this domain** and then enable JASPI. Select **Providers** to define providers for the domain.

Note: The JASPI authentication provider can be enabled with providers configured at the domain level. By default, all of the applications in the system have access to the JASPI authentication providers configured at the global level. The security runtime first checks for the JASPI authentication providers at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure JASPI authentication providers at a domain only when the provider is to be used exclusively by the applications in that security domain.

Authentication Mechanism Attributes

Specifies the various cache settings that need to applied at the domain level.

Select **Authentication cache settings** to specify your authentication cache settings. The configuration specified on this panel is applied only to this domain.

Select **LTPA Timeout** to configure a different LTPA timeout value at the domain level. The default timeout value is 120 minutes, which is set at the global level. If the LTPA timeout is set at the domain level, any token that is created in the security domain when accessing user applications is created with this expiration time.

When **Use realm-qualified user names** is enabled, user names returned by methods such as `getUserPrincipal( )` are qualified with the security realm (user registry) used by applications in the security domain.

Authorization Provider

You can configure an external third party JACC (Java Authorization Contract for Containers) provider at the domain level. Tivoli Access Manager's JACC provider can only be configured at

the global level. Security domains can still use it if they do not override the authorization provider with another JACC provider or with the built-in native authorization.

On z/OS, you can enable or disable the System Authorization Facility (SAF) based authorization at the domain level.

Custom properties

Set custom properties at the domain level that are either new or different from those at the global level. By default, all of the custom properties at the global security configuration can be accessed by all of the applications in the cell. The security runtime code first checks for the custom property at the domain level. If it does not find it, it then attempts to obtain the custom property from the global security configuration.

7. Click **Apply**.
8. After you have saved your configuration changes, restart the server for your changes to take effect.

Deleting multiple security domains

You can delete multiple security domains from your configuration. You must remove the resources assigned to the security domains before deleting them. Only remove those security domains that are not needed in your security configuration.

Before you begin

Only users assigned to the administrator role can delete security domains. Enable global security in your environment before deleting security domains.

Read about “Multiple security domains” on page 1222 for a better understanding of what multiple security domains are and how they are supported in this version of WebSphere Application Server.

About this task

Security domains provide a mechanism to use different security settings for administrative applications and user applications. They also provide the ability to support multiple security settings so different applications can use different security attributes like user registry or login configurations.

Perform the following steps to delete an existing security domain using the administrative console:

Note: Only delete the security domains after first removing any resources associated with them. The servers impacted should be restarted.

Procedure

1. Click **Security > Security domains**.
2. On the Security domains collection page, select a domain to delete.
3. Click **Delete**.

Copying multiple security domains

You can copy selected multiple security domains from the domain collection to create a new domain. This is useful if you want to create a domain that is similar to a previous domain. However, you might want to make a few slight adjustments. When copying an existing domain, you must supply a unique domain name for the new one.

Before you begin

Only users assigned to the administrator role can copy or create new multiple security domains. Enable global security in your environment before copying multiple security domains.

Read about “Multiple security domains” on page 1222 for a better understanding of what multiple security domains are and how they are supported in this version of WebSphere Application Server.

About this task

Security domains provide a mechanism to use different security settings for administrative applications and user applications. They also provide the ability to support multiple security settings so different applications can use different security attributes like user registry or login configurations.

Use multiple security domains to achieve the following goals:

- Configure different security attributes for administrative and user applications within a cell
- Consolidate server configurations by managing different security configurations within a cell
- Restrict access between applications with different user registries, or configure trust relationships between applications to support communication across registries

Perform the following steps to copy an existing security domain using the administrative console:

Procedure

1. Click **Security > Security domains**.
2. Optional: From Preferences, you can select the maximum number of rows to display when the domain collection is large. The default number of rows is 20. Rows that exceed that number appear on subsequent pages.
3. Select a domain to copy.
4. Click **Copy Selected Domain...** to copy an existing domain from the collection. You can optionally select **Copy Global Security..** to copy an existing domain and have it maintain its global security settings (collection selections are ignored). A new domain name is also required if you choose this option.
5. Specify a unique name for the domain. This field is required. A domain name must be unique within a cell and cannot contain an invalid character.
6. Specify a unique description for the domain.
7. Click **Apply**. After you click **Apply** you are returned to the Security domains detail page
8. Under Assigned Scopes, assign the security domain to the entire cell or select the specific servers, clusters, and service integration buses to include in the security domain.
9. Customize your security configuration by specifying security attributes for your new domain and by assigning it to cell resources.

   You can change security attributes such as the following:

   Application Security

   Specifies the settings for application security and Java 2 security. You can use the global security settings or customize the settings for a domain.

   Select **Enable application security** to enable or disable security this choice for user applications. When this selection is disabled, all of the EJBs and web applications in the security domain are no longer protected. Access is granted to these resources without user authentication. When you enable this selection, the J2EE security is enforced for all of the EJBs and web applications in the security domain. The J2EE security is only enforced when Global Security is enabled in the global security configuration, (that is, you cannot enable application security without first enabling Global Security at the global level).

   Java 2 Security

   Select **Java 2 security** to enable or disable Java 2 security at the domain level. This choice enables or disables Java 2 security at the process (JVM) level so that all applications (both administrative and user) can enable or disable Java 2 security.

   User realm

This section enables you to configure the user registry for the security domain. You can separately configure any registry that is used at the domain level. Read about “Multiple security domains” on page 1222 for more information.

Trust association

When you configure the trust association interceptor (TAI) at a domain level, the interceptors configured at the global level are copied to the domain level for convenience. You can modify the interceptor list at the domain level to fit your needs. Only configure those interceptors that are to be used at the domain level.

SPNEGO Web Authentication

The SPNEGO web authentication, which enables you to configure SPNEGO for web resource authentication, can be configured at the domain level.

Note: In WebSphere Application Server Version 6.1, a TAI that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. This function was deprecated in WebSphere Application Server 7.0. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

RMI/IIOP Security

The RMI/IIOP security attribute refers to the CSIv2 (Common Secure Interoperability version 2) protocol properties. When you configure these attributes at the domain level, the RMI/IIOP security configuration at the global level is copied for convenience.

You can change the attributes that need to be different at the domain level. The Transport layer settings for CSIv2 inbound communications should be the same for both the global and the domain levels. If they are different, the domain level attributes are applied to all of the application in the process.

JAAS application logins

Specifies the configuration settings for the Java Authentication and Authorization Service (JAAS) application logins. You can use the global security settings or customize the settings for a domain.

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. Be default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

JAAS system logins

Specifies the configuration settings for the JAAS system logins. You can use the global security settings or customize the configuration settings for a domain.

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. Be default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

JAAS J2C authentication

Specifies the configuration settings for the JAAS J2C authentication data. You can use the global security settings or customize the settings for a domain.

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. Be default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

Java Authentication SPI (JASPI)

Specifies the configuration settings for a Java Authentication SPI (JASPI) authentication provider. You can use the global security settings or customize the settings for a domain. To configure JASPI authentication providers for a domain, select **Customize for this domain** and then enable JASPI. Select **Providers** to define providers for the domain.

Note: The JASPI authentication provider can be enabled with providers configured at the domain level. By default, all of the applications in the system have access to the JASPI authentication providers configured at the global level. The security runtime first checks for the JASPI authentication providers at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure JASPI authentication providers at a domain only when the provider is to be used exclusively by the applications in that security domain.

Authentication Mechanism Attributes

Specifies the various cache settings that need to applied at the domain level.

Select **Authentication cache settings** to specify your authentication cache settings. The configuration specified on this panel is applied only to this domain.

Select **LTPA Timeout** to configure a different LTPA timeout value at the domain level. The default timeout value is 120 minutes, which is set at the global level. If the LTPA timeout is set at the domain level, any token that is created in the security domain when accessing user applications is created with this expiration time.

When **Use realm-qualified user names** is enabled, user names returned by methods such as `getUserPrincipal( )` are qualified with the security realm (user registry) used by applications in the security domain.

Authorization Provider

You can configure an external third party JACC (Java Authorization Contract for Containers) provider at the domain level. Tivoli Access Manager's JACC provider can only be configured at the global level. Security domains can still use it if they do not override the authorization provider with another JACC provider or with the built-in native authorization.

On z/OS, you can enable or disable the System Authorization Facility (SAF) based authorization at the domain level.

Custom properties

Set custom properties at the domain level that are either new or different from those at the global level. By default, all of the custom properties at the global security configuration can be accessed by all of the applications in the cell. The security runtime code first checks for the custom property at the domain level. If it does not find it, it then attempts to obtain the custom property from the global security configuration.

10. Click **Apply**.
11. After you have saved your configuration changes, restart the server for your changes to take effect.

Configuring inbound trusted realms for multiple security domains

You can configure which realms to grant inbound trust to for multiple security domains. The trust relationship between realms is used when communicating with Lightweight Third-Party Authentication (LTPA) tokens. Once a LTPA token is decrypted by the receiving server, the realm in the token is checked to see if it is trusted. If it is not, the validation of the token fails. A *realm* represents a user registry in WebSphere Application Server.

Before you begin

For information on cross realm communications, read the section in “Multiple security domains” on page 1222.

Only users assigned to the administrator role can configure multiple security domains. Enable global security in your environment before configuring multiple security domains.

Perform the following steps to grant inbound trusted realms for multiple security domains using the administrative console:

Procedure

1. Click **Security > Security domains**.
2. Select a domain to edit or create a new one. Under Security Attributes, click **User realm**.
3. Click **Customize for this domain**.
4. Under Related Items, select **Trusted authentication realms - inbound**.
5. Select **Trust all realms (including those external to this cell)** or **Trust realms as indicated below**. If Kerberos authentication is enabled, and you have cross realms or trusted realms, you must add the Kerberos trusted realm by selecting **Trust realms as indicated below**.
6. Click **Apply.**

What to do next

The realms you selected to trust accept messages from other trusted realms but do not accept messages from untrusted realms. Select **Add External Realm** to add trust for realms that are external to this cell.

Configure security domains

Use this page to configure the security attributes of a domain and to assign the domain to cell resources. For each security attribute, you can use the global security settings or customize settings for the domain.

To view this administrative console page, click **Security > Security domains**. On the Security domains collection page, select an existing domain to configure, create a new one, or copy an existing domain.

Read about “Multiple security domains” on page 1222 for a better understanding of what multiple security domains are and how they are supported in this version of WebSphere Application Server.

Name

Specifies a unique name for the domain. This name can not be edited after the initial submission.

A domain name must be unique within a cell and cannot contain an invalid character.

Description

Specifies a description for the domain.

Assigned Scopes

Select to display the cell topology. You can assign the security domain to the entire cell or select the specific clusters, nodes and service integration buses to include in the security domain.

If you select **All scopes**, the entire cell topology is displayed.

If you select **Assigned scopes**, the cell topology is displayed with those servers and clusters that are assigned to the current domain.

The name of an explicitly assigned domain appears next to any resource. Checked boxes indicate resources that are currently assigned to the domain. You also can select other resources and click **Apply** or **OK** to assign them to the current domain.

A resource that is not checked (disabled) indicates that it is not assigned to the current domain and must be removed from another domain before it can be enabled for the current domain.

If a resource does not have an explicitly-assigned domain, it uses the domain assigned to the cell. If no domain is assigned to the cell, then the resource uses global settings.

Cluster members cannot be individually assigned to domains; the enter cluster uses the same domain.

Application Security:

Select **Enable application security** to enable or disable security for user applications. You can use the global security settings or customize the settings for a domain.

When this selection is disabled, all of the EJBs and web applications in the security domain are no longer protected. Access is granted to these resources without user authentication. When you enable this selection, the J2EE security is enforced for all of the EJBs and web applications in the security domain. The J2EE security is only enforced when Global Security is enabled in the global security configuration, (that is, you cannot enable application security without first enabling Global Security at the global level).

Enable application security

Enables security for the applications in your environment. This type of security provides application isolation and requirements for authenticating application users

In previous releases of WebSphere Application Server, when a user enabled global security, both administrative and application security were enabled. In WebSphere Application Server Version 6.1, the previous notion of global security were split into administrative security and application security, each of which you can enable separately.

As a result of this split, WebSphere Application Server clients must know whether application security is disabled at the target server. Administrative security is enabled, by default. Application security is disabled, by default. To enable application security, you must enable administrative security. Application security is in effect only when administrative security is enabled.

When this selection is disabled, all of the EJBs and web applications in the security domain are no longer protected. Access is granted to these resources without user authentication. When you enable this selection, the J2EE security is enforced for all of the EJBs and web applications in the security domain. The J2EE security is only enforced when Global Security is enabled in the global security configuration, (that is, you cannot enable application security without first enabling Global Security at the global level).

Java 2 security:

Select **Use Java 2 security** to enable or disable Java 2 security at the domain level or to assign or add properties related to Java 2 security. You can use the global security settings or customize the settings for a domain.

This choice enables or disables Java 2 security at the process (JVM) level so that all applications (both administrative and user) can enable or disable Java 2 security.

Use global security settings
Select to specify the global security settings that are being used.

Customize for this domain
Select to specify the settings that are defined in the domain, such as options to enable application and Java 2 security and to use realm-qualified authentication data.

Use Java 2 security to restrict application access to local resources
Select to specify whether to enable or disable Java 2 security permission checking. By default, access to local resources is not restricted. You can choose to disable Java 2 security, even when application security is enabled.

When the **Use Java 2 security to restrict application access to local resources** option is enabled and if an application requires more Java 2 security permissions than are granted in the default policy, the application might fail to run properly until the required permissions are granted in either the `app.policy` file or the `was.policy` file of the application. AccessControl exceptions are generated by applications that do not have all the required permissions.

Warn if applications are granted custom permissions
Specifies that during application deployment and application start, the security runtime issues a warning if applications are granted any custom permissions. Custom permissions are permissions that are defined by the user applications, not Java API permissions. Java API permissions are permissions in the `java.*` and `javax.*` packages.

The application server provides support for policy file management. A number of policy files are available in this product, some of them are static and some of them are dynamic. Dynamic policy is a template of permissions for a particular type of resource. No code base is defined and no relative code base is used in the dynamic policy template. The real code base is dynamically created from the configuration and run-time data. The `filter.policy` file contains a list of permissions that you do not want an application to have according to the J2EE 1.4 specification.

Important: You cannot enable this option without enabling the **Use Java 2 security to restrict application access to local resources** option.

Restrict access to resource authentication data
This option is disabled if Java 2 security has not been enabled.

Consider enabling this option when both of the following conditions are true:

- Java 2 security is enforced.
- The application code is granted the accessRuntimeClasses WebSphereRuntimePermission permission in the `was.policy` file found within the application enterprise archive (EAR) file. For example, the application code is granted the permission when the following line is found in your `was.policy` file:

```
permission com.ibm.websphere.security.WebSphereRuntimePermission "accessRuntimeClasses";
```

The **Restrict access to resource authentication data** option adds fine-grained Java 2 security permission checking to the default principal mapping of the WSPrincipalMappingLoginModule implementation. You must grant explicit permission to Java 2 Platform, Enterprise Edition (J2EE) applications that use the WSPrincipalMappingLoginModule implementation directly in the Java Authentication and Authorization Service (JAAS) login when **Use Java 2 security to restrict application access to local resources** and the **Restrict access to resource authentication data** options are enabled.

Default: Disabled

User Realm:
This section enables you to configure the user registry for the security domain. You can separately configure any registry that is used at the domain level.

When configuring a registry at the domain level you can choose to define your own realm name for the registry. The realm name distinguishes one user registry from another. The realm name is used in multiple places – in the Java client login panel to prompt the user, in the authentication cache, and when using native authorization.

At the global configuration level, the system creates the realm for the user registry. In previous releases of WebSphere Application Server, only one user registry is configured in the system. When you have multiple security domains you can configure multiple registries in the system. For the realms to be unique in these domains, configure your own realm name for a security domain. You also can choose the system to create a unique realm name if it is certain to be unique. In the latter case, the realm name is based on the registry that is being used.

Trust Association:
Select to specify the settings for the trust association. Trust association is used to connect reversed proxy servers to the application servers.

Trust association enables the integration of IBM WebSphere Application Server security and third-party security servers. More specifically, a reverse proxy server can act as a front-end authentication server while the product applies its own authorization policy onto the resulting credentials that are passed by the proxy server.

Tivoli Access Manager's trust association interceptors can only be configured at the global level. The domain configuration can also use them, but cannot have a different version of the trust association interceptor. Only one instance of Tivoli Access Manager's trust association interceptors can exist in the system.

Note: The use of trust association interceptors (TAIs) for Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) authentication is deprecated. The SPNEGO web authentication panels provide a much easier way to configure SPNEGO.

Interceptors
Select to access or to specify the trust information for reverse proxy servers.

Enable trust association
Select to enable the integration of IBM WebSphere Application Server security and third-party security servers. More specifically, a reverse proxy server can act as a front-end authentication server while the product applies its own authorization policy onto the resulting credentials that are passed by the proxy server.

SPNEGO Web Authentication:
Specifies the settings for Simple and Protected GSS-API Negotiation (SPNEGO) as the web authentication mechanism.

The SPNEGO web authentication, which enables you to configure SPNEGO for web resource authentication, can be configured at the domain level.

Note: In WebSphere Application Server Version 6.1, a TAI that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

RMI/IIOP Security:

Specifies the settings for Remote Method Invocation over the Internet Inter-ORB Protocol (RMI/IIOP).

An Object Request Broker (ORB) manages the interaction between clients and servers, using the Internet InterORB Protocol (IIOP). It enables clients to make requests and receive responses from servers in a network-distributed environment.

When you configure these attributes at the domain level, the RMI/IIOP security configuration at the global level is copied for convenience. You can change the attributes that need to be different at the domain level. The Transport layer settings for CSIv2 inbound communications should be the same for both the global and the domain levels. If they are different, the domain level attributes are applied to all of the applications in the process.

When a process communicates with another process with a different realm, the LTPA authentication and the propagation tokens are propagated to the downstream server unless that server is listed in the outbound trusted realms list. This can be done using the **Trusted authentication realms – outbound** link on the **CSIv2 outbound communication** panel.

CSIv2 inbound communications

Select to specify authentication settings for requests that are received and transport settings for connections that are accepted by this server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

WebSphere Application Server enables you to specify Internet Inter-ORB Protocol (IIOP) authentication for both inbound and outbound authentication requests. For inbound requests, you can specify the type of accepted authentication, such as basic authentication.

CSIv2 outbound communications

Select to specify authentication settings for requests that are sent and transport settings for connections that are initiated by the server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

WebSphere Application Server enables you to specify Internet Inter-ORB Protocol (IIOP) authentication for both inbound and outbound authentication requests. For outbound requests, you can specify properties such as type of authentication, identity assertion or login configurations that are used for requests to downstream servers.

JAAS Application logins

Select to define login configurations that are used by JAAS.

The JAAS application logins, the JAAS system logins, and the JAAS J2C authentication data aliases can all be configured at the domain level. By default, all of the applications in the system have access to the JAAS logins configured at the global level. The security runtime first checks for the JAAS logins at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure any of these JAAS logins at a domain only when you need to specify a login that is used exclusively by the applications in the security domain.

For JAAS and custom properties only, once global attributes are customized for a domain they can still be used by user applications.

Do not remove the ClientContainer, DefaultPrincipalMapping, and WSLogin login configurations because other applications might use them. If these configurations are removed, other applications might fail.

Use global and domain-specific logins

Select to specify the settings that are defined in the domain, such as options to enable application and Java 2 security and to use realm-qualified authentication data.

JAAS System Logins:
Specifies the configuration settings for the JAAS system logins. You can use the global security settings or customize the configuration settings for a domain.

System Logins
Select to define the JAAS login configurations that are used by system resources, including the authentication mechanism, principal mapping, and credential mapping

JAAS J2C Authentication Data:
Specifies the settings for the JAAS J2C authentication data. You can use the global security settings or customize the settings for a domain.

Java 2 Platform, Enterprise Edition (J2EE) Connector authentication data entries are used by resource adapters and Java DataBase Connectivity (JDBC) data sources.

Use global and domain-specific entries
Select to specify the settings that are defined in the domain, such as options to enable application and Java 2 security and to use realm-qualified authentication data.

Java Authentication SPI (JASPI)
Specifies the configuration settings for a Java Authentication SPI (JASPI) authentication provider and associated authentication modules. You can use the global security settings or customize the settings for a domain. To configure JASPI authentication providers for a domain, select **Customize for this domain** and then you can enable JASPI. Select **Providers** to create or to edit a JASPI authentication provider.

Note: The JASPI authentication provider can be enabled with providers configured at the domain level. By default, all of the applications in the system have access to the JASPI authentication providers configured at the global level. The security runtime first checks for the JASPI authentication providers at the domain level. If it does not find them, it then checks for them in the global security configuration. Configure JASPI authentication providers at a domain only when the provider is to be used exclusively by the applications in that security domain.

Authentication Mechanism Attributes:
Specifies the various cache settings that must be applied at the domain level.

- Authentication cache settings - use to specify your authentication cache settings. The configuration specified on this panel is applied only to this domain.
- LTPA Timeout - You can configure a different LTPA timeout value at the domain level. The default timeout value is 120 minutes, which is set at the global level. If the LTPA timeout is set at the domain level, any token that is created in the security domain when accessing user applications is created with this expiration time.
- Use realm-qualified user names - When this selection is enabled, user names returned by methods such as `getUserPrincipal( )` are qualified with the security realm (user registry) used by applications in the security domain.

Authorization Provider:
Specifies the settings for the authorization provider. You can use the global security settings or customize the settings for a domain.

You can configure an external third party JACC (Java Authorization Contract for Containers) provider at the domain level. Tivoli Access Manager's JACC provider can only be configured at the global level. Security domains can still use it if they do not override the authorization provider with another JACC provider or with the built-in native authorization.

Select either the **Default authorization** or **External authorization using a JAAC provider**. The **Configure** button is only enabled when **External authorization using a JAAC provider** is selected.

z/OS security options:
Specifies the settings for z/OS. You can use the global security settings or customize the settings for a domain.

Custom properties
Select to specify name-value pairs of data, where the name is a property key and the value is a string.

Set custom properties at the domain level that are either new or different from those at the global level. By default, all of the custom properties at the global security configuration can be accessed by all of the applications in the system. The security runtime code first checks for the custom property at the domain level. If it does not find it, it then attempts to obtain the custom property from the global security configuration.

Web Services Bindings
Click **Default policy set bindings** to set the domain default provider and client bindings.

External realm name
Use this page to add a WebSphere Application Server realm that is external to this cell. The realm is initially not trusted. Use the Trusted authentication realms - inbound page to establish trust.

To view this administrative console page, click **Security > Security domains**. Select a domain to edit or create a new one. Under Security Attributes, click **User realm**. Click **Customize for this domain** and then select a **Realm type**. Click **Configure**. Under Related items, click **Trusted authentication realms - inbound** or **Trusted authentication realms - outbound**. Click **Add External Realm...**.

External realm name
Use to specify the name of the realm that is external to the list of realms that are available to receive trust.

Trust all realms
Use this page to configure which realms to grant inbound or outbound trust to.

The inbound trust is required to validate LTPA tokens that contain a foreign realm. The outbound trust is required to send the credential tokens to the trusted realms. For example, if an application using realmA needs to communicate using LTPA with an application using realmB, realmA should have realmB in its outbound trust list and realmB should have realmA in its inbound trust list.

To view this administrative console page, click **Security > Security domains**. Select a domain to edit or create a new one. Under Security Attributes, click **User realm**. Click **Customize for this domain**. Select a realm type and then click **Configure**.

Under Related items, click **Trusted authentication realms - inbound** or **Trusted authentication realms - outbound**.

Trust all realms (including those external to this cell)
Select to trust all of the realms listed on this page, including those external to the cell.

Trust realms as indicated below
Select to trust only those realms that you have selected from the list of realms that are available to receive inbound trust.

Add External Realm...
Select to add realms that are external to this cell to the list of realms that are available to receive inbound trust. When an external realm is added, it is trusted by default. If it is not trusted it is removed from the list.

Security domains collection

Security domains provide a mechanism to use different security settings for administrative applications and user applications. They also provide the ability to support multiple security settings so different application servers can use different security attributes like user registry or login configurations.

To view this administrative console page, click **Security > Security domains**.

Read about “Multiple security domains” on page 1222 for a better understanding of what multiple security domains are and how they are supported in this version of WebSphere Application Server.

Maximum rows

Specifies the maximum number of rows that display when the collection is large. The rows that are not displayed appear on the next page.

The default is 20. Rows that exceed the maximum number display on subsequent pages.

Retain filter criteria

Specifies whether to use the same filter criteria entered in the show filter function to display this page the next time you visit it.

Copy selected domain

Select to copy a selected domain from the collection (a new name is required)

Copy global security

Select to create a domain with a copy of the global security settings (collection selections are ignored). A domain name is required.

Authentication cache settings

Use this page to specify your authentication cache settings.

To view this administrative console page, click **Security > Global security > Authentication cache settings**.

Enable authentication cache

Specifies whether to disable the authentication cache.

Leave the authentication cache enabled for performance reasons. However, you can disable the authentication cache for debug or measurement purposes. When this choice is disabled, the performance is impacted since whenever a user is authenticated the user registry is accessed to gather information about the user. New tokens are then created for the user.

Default: Enabled

Cache timeout:

Specifies the time period at which the authenticated credential in the cache expires. Verify that this time period is less than the value for the **Timeout value for forwarded credentials between servers** field (the LTPA timeout).

If the application server infrastructure security is enabled, the security cache timeout can influence performance. The timeout setting specifies how often to refresh the security-related caches. Security information pertaining to beans, permissions, and credentials is cached. When the cache timeout expires, all cached information not accessed within the timeout period is purged from the cache. Subsequent requests for the information result in a database lookup. On occasion, acquiring the information requires

invoking a Lightweight Directory Access Protocol (LDAP)-bind or native authentication. Both invocations are relatively costly operations for performance. Determine the best trade-off for the application by looking at usage patterns and security needs for the site.

You must consider the following effects of this value on your configuration:

- Larger authentication cache timeout values can increase the security risk. For example, you might revoke a user in the user registry or repository. However, the revoked user can log into the administrative console using the credential that is cached in the authentication cache until the cache is refreshed.
- Smaller authentication cache timeout values can affect performance. When this value is smaller, the application server accesses the user registry or repository more frequently.
- Larger numbers of entries in the authentication cache, which is due to an increased number of users, increases the memory usage by the authentication cache. Thus, the application server might slow down and affect performance.

You can limit the size of the authentication cache by setting the maximum cache size value. Set both the maximum cache size and the authentication cache timeout values to balance your security risk and performance needs.

The LTPA timeout value should not be set lower than the security cache timeout value. The LTPA timeout value should be set higher than the ORB request timeout value. However, there is no relation between the security cache timeout value and the ORB request timeout value. For more information on the LTPA timeout value, see the documentation about authentication mechanisms and expiration. For more information on the ORB request timeout value, see the documentation about the Object Request Broker service settings.

Default: 10 minutes

Initial cache size:

Specifies the initial size of the hash table caches.

A higher number of available hash values might decrease the occurrence of hash collisions. A hash collision results in a linear search for the hash bucket, which might decrease the retrieval time. If several entries compose a hash table cache, create a table with a larger capacity that supports more efficient hash entries instead of allowing automatic rehashing determine the growth of the table. Rehashing causes every entry to move each time.

Default: 50

Maximum cache size

Indicates the maximum size of the cache.

After this limit is reached, the least used entries are removed from the cache to make space for the new entries.

Default: 25000

Use basic authentication cache keys (password one-way hashed):

Caches the userName and the one-way hashed password as the key lookup in the cache.

Disable this only if you do not want this information to be stored in the cache. If this is disabled, every time a user logs in with userName and password, the user registry is accessed, which impacts performance.

Default: True

Authenticating users

The process of authenticating users involves a user registry and an authentication mechanism. Optionally, you can define trust between WebSphere Application Server and a proxy server, configure single sign-on capability, and specify how to propagate security attributes between application servers.

About this task

The following security topics are covered in this section:

User registries
For information on local operating system, Lightweight Directory Access Protocol (LDAP), custom user registries, and user repositories such as virtual member manager, see "Selecting a registry or repository."

Trust associations
For more information on trust associations, see "Trust associations" on page 1433.

Single sign-on
For more information on single sign-on, see "Single sign-on for authentication using LTPA cookies" on page 1438.

Security attribute propagation
For more information on propagation tokens, authorization tokens, single sign-on tokens, and authentication tokens, see "Security attribute propagation" on page 1544.

The following information is covered in this section:

Procedure

- Configure a user registry. For more information, see "Selecting a registry or repository."
- Configure WebSEAL or a custom trust association interceptor. For more information see, "Integrating third-party HTTP reverse proxy servers" on page 1433.
- Configure single sign-on. For more information, see "Implementing single sign-on to minimize web user authentications" on page 1441.
- Propagate security attributes. For more information, see "Propagating security attributes among application servers" on page 1549.
- Configure the authentication cache. For more information, see "Configuring the authentication cache" on page 1560.

What to do next

After completing the configuring the authentication process, you must authorize access to resources. For more information, see "Authorizing access to resources" on page 1614.

Selecting a registry or repository

Information about users and groups reside in a user registry. In WebSphere Application Server, a user registry authenticates a user and retrieves information about users and groups to perform security-related functions, including authentication and authorization.

Before you begin

Note: During profile creation, either during installation or post-installation, administrative security is enabled by default. The file-based federated user repository is configured as the active user registry. Decide if you want a different user registry.

Before configuring the user registry or repository, decide which user registry or repository to use. You can configure one Active default registry for the Cell.

About this task

WebSphere Application Server provides implementations that support multiple types of registries and repositories including the local operating system registry, a stand-alone Lightweight Directory Access Protocol (LDAP) registry, a stand-alone custom registry, and federated repositories.

With WebSphere Application Server, a user registry or a repository, such as a federated repository, authenticates a user and retrieves information about users and groups to perform security-related functions including authentication and authorization.

With WebSphere Application Server, a user registry or repository is used for:

- Authenticating a user using basic authentication, identity assertion, or client certificates
- Retrieving information about users and groups to perform security-related administrative functions, such as mapping users and groups to security roles

In addition to local operating system, LDAP, and Federated repository registries, WebSphere Application Server also provides a plug-in to support any registry by using the custom registry feature. The custom registry feature enables you to configure any user registry that is not made available through the security configuration panels of the WebSphere Application Server.

Configuring the correct registry or repository is a prerequisite to assigning users and groups to roles for applications. When a user registry or repository is not configured, the local operating system registry is used by default. If your choice of user registry is not the local operating system registry, you need to first configure the registry or repository, which is normally done as part of enabling security, restart the servers, and then assign users and groups to roles for all your applications.

WebSphere Application Server supports the following types of user registries:

- Federated repository
- Local operating system
- Standalone Lightweight Directory Access Protocol (LDAP) registry
- Stand-alone custom registry

The UserRegistry interface is used to implement both the custom registry and the federated repository options for the user account repository. The interface is very helpful in situations where the current user and group information exists in some other formats, for example, a database, and cannot move to local operating system or LDAP registries. In such a case, you can implement the UserRegistry interface so that WebSphere Application Server can use the existing registry for all the security-related operations. The process of implementing a custom registry is a software implementation effort, and it is expected that the implementation does not depend on WebSphere Application Server resource management for its operation. For example, you cannot use an Application Server data source configuration; generally you must invoke database connections and dictate their behavior directly in your code.

Note: WebSphere Application Server has implemented a user registry proxy by using the UserRegistry interface. However, the return values are little different from the interface. For example, getUniqueUserId returns the uniqueID with the realm name wrapped. You cannot use the return value to pass to getUserSecurityName, as shown in the following example:

```
// Retrieves the default InitialContext for this server.
javax.naming.InitialContext ctx = new javax.naming.InitialContext();

// Retrieves the local UserRegistry object.
com.ibm.websphere.security.UserRegistry reg =
        (com.ibm.websphere.security.UserRegistry) ctx.lookup("UserRegistry");

// Retrieves the registry uniqueID based on the userName that is specified
```

```
    // in the NameCallback.
String uniqueid = reg.getUniqueUserId(userName);
// Strip the realm name and get real uniqueID
String uid = com.ibm.wsspi.security.token.WSSecurityPropagationHelper.getUserFromUniqueID (uniqueID);

// Retrieves the security name from the user registry based on the uniqueID.
String securityName = reg.getUserSecurityName(uid);
```

You can use a Service Provider Interface (SPI) for this parsing function.

After the applications are assigned users and groups and you need to change the user registries, delete all the users and groups, including any RunAs role, from the applications, and reassign them after changing the registry through the administrative console or by using wsadmin scripting. The following **wsadmin** command, which uses Jacl, removes all of the users and groups from any application:

```
$AdminApp deleteUserAndGroupEntries yourAppName
```

where *yourAppName* is the name of the application. Backing up the old application is advised before performing this operation. However, if both of the following conditions are true, you might be able to switch the registries without having to delete the users and groups information:

- All of the user and group names, including the password for the RunAs role users, in all of the applications match in both user registries.
- The application bindings file does not contain the access IDs which are unique for each user registry even for the same user or group name.

By default, an application does not contain access IDs in the bindings file. These IDs are generated when the applications start. However, if you migrated an existing application from an earlier release, or if you used the wsadmin script to add access IDs for the applications to improve performance, you have to remove the existing user and group information and add the information after configuring the new user registry.

For more information on updating access IDs, see updateAccess IDs in the Commands for the AdminApp object article.

Attention: WebSphere Application Server supports a variety of user registries and repositories on different operating systems. During the user authentication process, you might use non-alphanumeric characters in your user name or password. Restrictions on the use of these non-alphanumeric characters depends on both the underlying operating system and the user registry type. For more information on which non-alphanumeric characters are not supported, see your operating system and user registry or repository documentation.

For a comprehensive list of the non-alphanumeric characters that are not supported, see the IBM AIX operating system documentation.

Complete one of the following steps to configure your user registry:

Procedure

- “Configuring local operating system registries” on page 1257
- “Configuring Lightweight Directory Access Protocol user registries” on page 1260
- “Configuring stand-alone custom registries” on page 1287.
- “Managing the realm in a federated repository configuration” on page 1316

What to do next

1. If you are enabling security, make sure that you complete the remaining steps. Verify that the User account repository on the Global security panel is set to the appropriate registry or repository. As the final step, validate the user ID and the password by clicking **Apply** on the Global security panel. Save, stop and start all WebSphere Application Servers.

2. For any changes in user registry panels to be effective, you must validate the changes by clicking **Apply** on the Global security panel. After validation, save the configuration and stop and start all WebSphere Application Servers, including the cells, nodes and all of the application servers. To avoid inconsistencies between the WebSphere Application Server processes, make sure that any changes to the registry or repository are done when all of the processes are running. If any of the processes are down, force synchronization to make sure that the process can start later.

   If the server or servers start without any problems, the setup is correct.

Configuring local operating system registries

Use these steps to configure local operating system registries.

Before you begin

For detailed information about using the local operating system user registry, see “Local operating system registries” on page 1258. These steps set up security based on the local operating system user registry on which WebSphere Application Server is installed.

In WebSphere Application Server Version 6.1, you can use an internally-generated server ID because the Security WebSphere Common Configuration Model (WCCM) model contains a new tag, internalServerId. You do not need to specify a server user ID and a password during security configuration except in a mixed-cell environment. See “Administrative roles and naming service authorization” on page 1615 for more detailed information about the new internal server ID.

About this task

The following steps are needed to perform this task initially when setting up security for the first time.

Procedure

1. Click **Security > Global security**.
2. Under User account repository, select **Local operating system** and click **Configure**.
3. Enter a valid user name in the **Primary administrative user name** field. This value is the name of a user with administrative privileges that is defined in the registry. This user name is used to access the administrative console or used by wsadmin.
4. Click **Apply**.
5. Select either the **Automatically generated server identity** or **Server identity that is stored in the repository** option. If you select the **Server identity that is stored in the repository** option, enter the following information:

   Server user ID or administrative user on a Version 6.0.x node
   Specify the short name of the account that is chosen in the second step.

   Server user password
   Specify the password of the account that is chosen in the second step.
6. Enter a valid user profile name in the **Primary administrative user name** field.

   The Primary administrative user name specifies the user profile to use when the server authenticates to the underlying operating system. This identity is also the user that has initial authority to access the administrative application through the administrative console. The administrative user ID is common to all user registries. The administrative ID is a member of the chosen registry and it has special privileges in WebSphere Application Server. However, it does not have any special privileges in the registry that it represents. In other words, you can select any valid user ID in the registry to use as the administrative user ID or server user ID.

   For the **Primary administrative user name** field, you can specify any user profile that meets this criteria:

   - The user profile has a status of `*ENABLED`.

- The user profile has a valid password.
- The user profile is not used as a group profile.

  Important: A group profile is assigned a unique group ID number, which is not assigned to a regular user profile. Run the DSPUSRPRF Display User Profile command to determine if the user profile you want to use as the Primary administrative user name has a defined group ID number. If the **Group ID** field is set to *NONE, you can use the user profile as the Primary administrative user name.

7. Click **OK**.

   The administrative console does not validate the user ID and password when you click **OK**. Validation is only done when you click **OK** or **Apply** in the Global security panel. First, make sure that you select **Local operating system** as the available realm definition in the User account repository section, and click **Set as current**. If security was already enabled and you had changed either the user or the password information in this panel, make sure to go to the Global security panel and click **OK** or **Apply** to validate your changes. If your changes are not validated, the server might not start.

   Important: Until you authorize other users to perform administrative functions, you can only access the administrative console with the server user ID and password that you specified. For more information, see “Authorizing access to administrative roles” on page 1679.

Results

For any changes in this panel to be effective, you need to save, stop, and start all the product servers, including nodes and application servers. If the server comes up without any problems, the setup is correct.

After completed these steps, you have configured WebSphere Application Server to use the local operating system registry to identify authorized users.

What to do next

Complete any remaining steps for enabling security. For more information, see “Enabling security” on page 1172.

Local operating system registries:

With the registry implementation for the local operating system, the WebSphere Application Server authentication mechanism can use the user accounts database of the local operating system.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log , SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

If you want to use the local operating system registry to represent the principals who access your WebSphere Application Server resources, you do not have to complete any special user registry setup steps. The local operating system registry is used for authentication and authorization of users who access WebSphere Application Server resources, but not for WebSphere Application Server users who access operating system resources. WebSphere Application Server does not run under the operating system user profile of Application Server users. Instead, WebSphere Application Server runs under the operating system profile that is configured by the Application Server administrator.

If you want to authorize a user for any WebSphere Application Server resource, a user profile for that user must exist in the operating system. Use the Create User Profile (CRTUSRPRF) command to create new user IDs that can be used by WebSphere Application Server

Do not use a local operating system registry in a WebSphere Application Server environment where application servers are dispersed across more than one machine because each machine has its own user registry.

As mentioned previously, the access IDs taken from the user registry are used during authorization checks. Because these IDs are typically unique identifiers, they vary from machine to machine, even if the exact users and passwords exist on each machine.

Web client certificate authentication is not currently supported when using the local operating system user registry. However, Java client certificate authentication does function with a local operating user registry. Java client certificate authentication maps the first attribute of the certificate domain name to the user ID in the user registry.

Even though Java client certificates function correctly, the following error displays in the `SystemOut.log` file:

```
CWSCJ0337E: The mapCertificate method is not supported
```

The error is intended for web client certificates; however, it also displays for Java client certificates. Ignore this error for Java client certificates.

Using either the local or the domain user registry

. If you want to access users and groups from either the local or the domain user registry, instead of both, set the com.ibm.websphere.registry.UseRegistry property. This property can be set to either `local` or `domain`. When this property is set to `local` (case insensitive) only the local user registry is used. When this property is set to domain, (case insensitive) only the domain user registry is used.

Set this property by completing the following steps to access the **Custom Properties** panel in the administrative console:

1. Click **Security > Global security**
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Local operating system**, and click **Configure**.
3. Under Additional properties, click **Custom properties**.

You can also use wsadmin to configure this property. When the property is set, the privilege requirement for the user who is running the product process does not change. For example, if this property is set to `local`, the user that is running the process requires the same privilege, as if the property was not set.

Using system user registries

The following notes apply when you use system user registries:

Local operating system settings:

Use this page to configure local operating system registry settings.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Local operating system**.
3. Click **Configure**.

WebSphere Application Server Version 7.0 distinguishes between the user identities for administrators who manage the environment and server identities for authenticating server to server communications. In most cases, server identities are automatically generated and are not stored in a repository.

Primary administrative user name:

Specifies the name of a user with administrative privileges that is defined in your local operating system.

The user name is used to log on to the administrative console when administrative security is enabled..

Attention: In WebSphere Application Server, Version 6.1 and above, a single user identity is required for both administrative access and internal process communication. When migrating to Version 6.1 and above, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Automatically generated server identity:

Enables the application server to generate the server identity, which is recommended for environments that contain only Version 6.1 or later nodes. Automatically generated server identities are not stored in a user repository.

Default: Enabled

Server identity that is stored in the repository:

Specifies a user identity in the repository that is used for internal process communication. Cells that contain Version 6.1 or later nodes require a server user identity that is defined in the active user repository.

Default: Enabled

Local operating system wizard settings:

Use this security wizard page to configure local operating system registry settings.

To view this security wizard page, complete the following steps:

1. Click **Security > Global security > Security configuration wizard**.
2. Select your protection settings and click **Next**.
3. Select the **Local operating system** option and click **Next**.

Primary administrative user name:

Specifies the name of a user with administrative privileges that is defined in your local operating system.

The user name is used to log on to the administrative console when administrative security is enabled..

Attention: In WebSphere Application Server, Version 6.1 and above, a single user identity is required for both administrative access and internal process communication. When migrating to Version 6.1 and above, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Configuring Lightweight Directory Access Protocol user registries

To access a user registry using the Lightweight Directory Access Protocol (LDAP), you must know a valid user name (ID) and password, the server host and port of the registry server, the base distinguished name

(DN) and, if necessary, the bind DN and the bind password. You can choose any valid user in the user registry that is searchable. You can use any user ID that has the administrative role to log in.

Before you begin

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

There are two different identities that are used for security purposes: the user ID for administrative functions and the server identity. When administrative security is enabled, the user ID and password for administrative functions is authenticated with the registry. If authentication fails, access to the administrative console is not granted or tasks with wsadmin scripts are not completed. It is important to choose an ID and password that do not expire or change often. If this user ID or password need to change in the registry, make sure that the changes are performed when all the application servers are up and running. When changes are to be made in the registry, review the article on “Standalone Lightweight Directory Access Protocol registries” on page 1407 (LDAP) before beginning this task.

The server identity is used for internal process communication. As part of this task, you can change the server identity from the default automatically generated ID to a server ID and password from the LDAP repository.

Procedure

1. In the administrative console, click **Security** > **Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.
3. Enter a valid user name in the **Primary administrative user name** field. Typically, the user name is the short name of the user and is defined by the user filter in the Advanced LDAP settings panel.
4. Determine whether to specify the user identity that is used for internal process communication. Cells that contain Version 5.1 or 6.x nodes require a server user identity that is defined in the active user repository. By default, the **Automatically generated server identity** option is enabled, and the application server generates the server identity. However, you can select the **Server identity that is stored in the repository** option to specify both the server identity and its associated password.
5. Select the type of LDAP server to use from the **Type** list. The type of LDAP server determines the default filters that are used by WebSphere Application Server. These default filters change the **Type** field to **Custom**, which indicates that custom filters are used. This action occurs after you click **OK** or **Apply** in the Advanced LDAP settings panel. Choose the **Custom** type from the list and modify the user and group filters to use other LDAP servers, if required.

   IBM Tivoli Directory Server users can choose IBM Tivoli Directory Server as the directory type. Use the IBM Tivoli Directory Server directory type for better performance. For a list of supported LDAP servers, see the Supported hardware, software, and APIs website.

   Attention: IBM SecureWay™ Directory Server has been renamed to IBM Tivoli Directory Server in WebSphere Application Server version 6.1.
6. Enter the fully qualified host name of the LDAP server in the **Host** field. You can enter either the IP address or domain name system (DNS) name.
7. Enter the LDAP server port number in the **Port** field. The host name and the port number represent the realm for this LDAP server in the WebSphere Application Server cell. So, if servers in different cells are communicating with each other using Lightweight Third Party Authentication (LTPA) tokens, these realms must match exactly in all the cells.

The default value is 389. If multiple WebSphere Application Servers are installed and configured to run in the same single sign-on domain, or if the WebSphere Application Server interoperates with a previous version of the WebSphere Application Server, then it is important that the port number match all configurations. For example, if the LDAP port is explicitly specified as 389 in a version 5.*x* configuration, and a WebSphere Application Server at version 6.0.*x* is going to interoperate with the version 5.*x* server, then verify that port 389 is specified explicitly for the version 6.0.*x* server.

You can set the com.ibm.websphere.security.ldap.logicRealm custom property to change the value of the realm name that is placed in the token. For more information, see the security custom properties topic.

8. Enter the base distinguished name (DN) in the **Base distinguished name** field. The base DN indicates the starting point for searches in this LDAP directory server. For example, for a user with a DN of cn=John Doe, ou=Rochester, o=IBM, c=US, specify the base DN as any of the following options, assuming a suffix of c=us:
   - ou=Rochester, o=IBM, c=us
   - o=IBM, c=us
   - c=us

   For authorization purposes, this field is case sensitive by default. Match the case in your directory server. If a token is received (for example, from another cell or Lotus Domino) the base DN in the server must match exactly the base DN from the other cell or Domino. If case sensitivity is not a consideration for authorization, enable the **Ignore case for authorization** option.

   In WebSphere Application Server, the distinguished name is normalized according to the Lightweight Directory Access Protocol (LDAP) specification. Normalization consists of removing spaces in the base distinguished name before or after commas and equal symbols. An example of a non-normalized base distinguished name is o = ibm, c = us or o=ibm, c=us. An example of a normalized base distinguished name is o=ibm,c=us.

   To interoperate between WebSphere Application Server Version 6.0 and later versions, you must enter a normalized base distinguished name in the **Base Distinguished Name** field. In WebSphere Application Server, Version 6.0 or later, the normalization occurs automatically during runtime.

   This field is required for all LDAP directories except the Lotus Domino Directory. The **Base Distinguished Name** field is optional for the Domino server.
9. Optional: Enter the bind DN name in the **Bind distinguished name** field. The bind DN is required if anonymous binds are not possible on the LDAP server to obtain user and group information. If the LDAP server is set up to use anonymous binds, leave this field blank. If a name is not specified, the application server binds anonymously. See the **Base Distinguished Name** field description for examples of distinguished names.
10. Optional: Enter the password corresponding to the bind DN in the **Bind password** field.
11. Optional: Modify the Search time out value. This timeout value is the maximum amount of time that the LDAP server waits to send a response to the product client before stopping the request. The default is 120 seconds.
12. Ensure that the **Reuse connection** option is selected. This option specifies that the server should reuse the LDAP connection. Clear this option only in rare situations where a router is used to send requests to multiple LDAP servers and when the router does not support affinity. Leave this option selected for all other situations.
13. Optional: Verify that the **Ignore case for authorization** option is enabled. When you enable this option, the authorization check is case insensitive. Normally, an authorization check involves checking the complete DN of a user, which is unique in the LDAP server and is case sensitive. However, when you use either the IBM Directory Server or the Sun ONE (formerly iPlanet) Directory Server LDAP servers, you must enable this option because the group information that is obtained from the LDAP servers is not consistent in case. This inconsistency affects the authorization check only. Otherwise, this field is optional and can be enabled when a case sensitive authorization check is required. For example, you might select this option when you use certificates and the certificate contents do not match the case of the entry in the LDAP server.

You can also enable the **Ignore case for authorization** option when you are using single sign-on (SSO) between the product and Lotus Domino. The default is enabled.

14. Optional: Select the **SSL enabled** option if you want to use Secure Sockets Layer communications with the LDAP server.

    Important: This step will only be successful provided that the Signer certificate for the LDAP is first added to the truststore that will be eventually used. If the Signer certificate from the LDAP is not added to the truststore, then

    - An error will be issued by the Administrative console.
    - the deployment manager (DMGR) systemout.log will show the **CWPKI0022E: SSL HANDSHAKE FAILURE** message indicating that the Signer certificate needs to be added to the truststore.

    To ensure an error free operation for this step, You need to first extract to a file the Signer certificate of the LDAP and send that file to the WebSphere Application Server machine. You can then add the certificate to the truststore being defined for the LDAP. In this way, you are assured that the remaining actions for this step will be successful.

    If you select the **SSL enabled** option, you can select either the **Centrally managed** or the **Use specific SSL alias** option.

    Centrally managed

    Enables you to specify an SSL configuration for particular scope such as the cell, node, server, or cluster in one location. To use the **Centrally managed** option, you must specify the SSL configuration for the particular set of endpoints. The Manage endpoint security configurations and trust zones panel displays all of the inbound and outbound endpoints that use the SSL protocol. If you expand the Inbound or Outbound section of the panel and click the name of a node, you can specify an SSL configuration that is used for every endpoint on that node. For an LDAP registry, you can override the inherited SSL configuration by specifying an SSL configuration for LDAP. To specify an SSL configuration for LDAP, complete the following steps:

    a. Click **Security > SSL certificate and key management > Manage endpoint security configurations and trust zones**.
    b. Expand **Outbound** > *cell_name* > **Nodes** > *node_name* > **Servers** > *server_name* > **LDAP**.

    Use specific SSL alias

    Select the **Use specific SSL alias** option if you intend to select one of the SSL configurations in the menu below the option.

    This configuration is used only when SSL is enabled for LDAP. The default is *DefaultSSLSettings*. You can click the name of an existing configuration to modify it or complete the following steps to create a new SSL configuration:

    a. Click **Security > SSL certificate and key management**.
    b. Under Configuration settings, click **Manage endpoint security configurations**.
    c. Select a Secure Sockets Layer (SSL) *configuration_name* for selected scopes, such as a cell, node, server, or cluster.
    d. Under Related items, click **SSL configurations**.
    e. Click **New**.

15. Click **OK** and either **Apply** or **Save** until you return to the Global security panel.

Results

This set of steps is required to set up the LDAP user registry. This step is required as part of enabling security in the WebSphere Application Server.

What to do next

1. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186.
2. Save, stop, and restart all the product servers (deployment managers, nodes and Application Servers) for changes in this panel to take effect. If the server comes up without any problems the setup is correct.

Standalone LDAP registry settings:

Use this page to configure Lightweight Directory Access Protocol (LDAP) settings when users and groups reside in an external LDAP directory.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.

When security is enabled and any of these properties change, go to the Global security panel and click **Apply** to validate the changes.

WebSphere Application Server Version 7.0 distinguishes between the user identities for administrators who manage the environment and server identities for authenticating server to server communications. In most cases, server identities are automatically generated and are not stored in a repository.

Note: The initial profile creation configures WebSphere Application Server to use a federated repositories security registry option with the file-based registry. This security registry configuration can be changed to use other options, including the stand-alone LDAP registry. Instead of changing from the federated repositories option to the stand-alone LDAP registry option under the User account repository configuration, consider employing the federated repositories option, which provides for LDAP configuration. Federated repositories provide a wide range of capabilities, including the ability to have one or multiple user registries. It supports federating one or more LDAPs in addition to file-based and custom registries. It also has improved failover capabilities, and a robust set of member (user and group) management capabilities. Federated repositories is required when you are using the new member management capabilities in WebSphere Portal 6.1 and above, and Process Server 6.1 and above. The use of federated repositories is required for following LDAP referrals, which is a common requirement in some LDAP server environments (such as Microsoft Active Directory).

It is recommended that you migrate from stand-alone LDAP registries to federated repositories. If you move to WebSphere Portal 6.1 and above, and or WebSphere Process Server 6.1 and above, you should migrate to federated repositories prior to these upgrades. For more information about federated repositories and its capabilities, read the Federated repositories topic. For more information about how to migrate to federated repositories, read the Migrating a stand-alone LDAP repository to a federated repositories LDAP repository configuration topic.

Primary administrative user name:

Specifies the name of a user with administrative privileges that is defined in your user registry.

The user name is used to log onto the administrative console when administrative security is enabled. Versions 6.1 and later require an administrative user that is distinct from the server user identity so that administrative actions can be audited.

Attention: In WebSphere Application Server, Version 6.0.x, a single user identity is required for both administrative access and internal process communication. When you migrate to Version 7.0, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Automatically generated server identity:

Enables the application server to generate the server identity, which is recommended for environments that contain only Version 6.1 or later nodes. Automatically generated server identities are not stored in a user repository.

Default: Enabled

Server identity that is stored in the repository:

Specifies a user identity in the repository that is used for internal process communication. Cells that contain Version 6.1 or later nodes require a server user identity that is defined in the active user repository.

Default: Enabled

Type of LDAP server:

Specifies the type of LDAP server to which you connect.

IBM SecureWay Directory Server is not supported.

Host:

Specifies the host ID (IP address or domain name service (DNS) name) of the LDAP server.

Port:

Specifies the host port of the LDAP server.

If multiple application servers are installed and configured to run in the same single sign-on domain or if the application server interoperates with a previous version, it is important that the port number match all configurations. For example, if the LDAP port is explicitly specified as `389` in a Version 6.1 configuration, and a WebSphere Application Server at Version 7.0 is going to interoperate with the Version 6.1 server, verify that port `389` is specified explicitly for the Version 7.0 server.

Default: 389
Type: Integer

Base distinguished name (DN):

Specifies the base distinguished name (DN) of the directory service, which indicates the starting point for LDAP searches of the directory service. In most cases, bind DN and bind password are needed. However, when anonymous bind can satisfy all of the required functions, bind DN and bind password are not needed.

For example, for a user with a DN of `cn=John Doe , ou=Rochester, o=IBM, c=US`, specify the Base DN as any of the following options: `ou=Rochester, o=IBM, c=US` or `o=IBM c=US` or `c=US`. For authorization purposes, this field is case sensitive. This specification implies that if a token is received, for example, from another cell or Lotus Domino, the base DN in the server must match the base DN from the other cell or Lotus

Domino server exactly. If case sensitivity is not a consideration for authorization, enable the **Ignore case for authorization** option. This option is required for all Lightweight Directory Access Protocol (LDAP) directories, except for the Lotus Domino Directory, IBM Tivoli Directory Server V6.0, and Novell eDirectory, where this field is optional.

Bind distinguished name (DN):

Specifies the DN for the application server to use when binding to the directory service.

If no name is specified, the application server binds anonymously. See the Base distinguished name (DN) field description for examples of distinguished names.

Bind password:

Specifies the password for the application server to use when binding to the directory service.

Search timeout:

Specifies the timeout value in seconds for a Lightweight Directory Access Protocol (LDAP) server to respond before stopping a request.

Default: 120

Reuse connection:

Specifies whether the server reuses the LDAP connection. Clear this option only in rare situations where a router is used to distribute requests to multiple LDAP servers and when the router does not support affinity.

Default: Enabled
Range: Enabled or Disabled

Important: Disabling the **Reuse connection** option causes the application server to create a new LDAP connection for every LDAP search request. This situation impacts system performance if your environment requires extensive LDAP calls. This option is provided because the router is not sending the request to the same LDAP server. The option is also used when the idle connection timeout value or firewall timeout value between the application server and LDAP is too small.

If you are using WebSphere Edge Server for LDAP failover, you must enable TCP resets with the Edge server. A TCP reset causes the connection to immediately closed and a backup server to failover. For more information, see "Sending TCP resets when server is down" at http://www.ibm.com/software/webservers/appserv/doc/v50/ec/infocenter/edge/LBguide.htm#HDRRESETSERVER and the Edge Server V2 - TCP Reset feature in PTF #2 described in: ftp://ftp.software.ibm.com/software/websphere/edgeserver/info/doc/v20/en/updates.pdf.

Ignore case for authorization:

Specifies that a case insensitive authorization check is performed when using the default authorization.

This option is required when IBM Tivoli Directory Server is selected as the LDAP directory server.

This option is required when Sun ONE Directory Server is selected as the LDAP directory server. For more information, see "Using specific directory servers as the LDAP server" in the documentation.

This option is optional and can be enabled when a case-sensitive authorization check is required. For example, use this option when the certificates and the certificate contents do not match the case that is used for the entry in the LDAP server. You can enable the **Ignore case for authorization** option when using single sign-on (SSO) between the application server and Lotus Domino.

Default: Enabled
Range: Enabled or Disabled

SSL enabled:

Specifies whether secure socket communication is enabled to the Lightweight Directory Access Protocol (LDAP) server.

When enabled, the LDAP Secure Sockets Layer (SSL) settings are used, if specified.

Centrally managed:

Specifies that the selection of an SSL configuration is based upon the outbound topology view for the Java Naming and Directory Interface (JNDI) platform.

Centrally managed configurations support one location to maintain SSL configurations rather than spreading them across the configuration documents.

Default: Enabled

Use specific SSL alias:

Specifies the SSL configuration alias to use for LDAP outbound SSL communications.

This option overrides the centrally managed configuration for the JNDI platform.

Standalone LDAP registry wizard settings:

Use this security wizard page to provide the basic settings to connect the application server to an existing Lightweight Directory Access Protocol (LDAP) registry.

To view this security wizard page, click **Security > Global security > Security configuration wizard**. You can modify your LDAP registry configuration by completing the following steps:

1. Click **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select**Standalone LDAP registry**, and click **Configure**.

Primary administrative user name:

Specifies the name of a user with administrative privileges that is defined in your user registry.

The user name is used to log onto the administrative console when administrative security is enabled. Versions 6.1 and later require an administrative user that is distinct from the server user identity so that administrative actions can be audited.

Attention: In WebSphere Application Server, Version 6.0.x, a single user identity is required for both administrative access and internal process communication. When you migrate to Version 7.0, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Type of LDAP server:

Specifies the type of LDAP server to which you connect.

IBM SecureWay Directory Server is not supported.

Host:

Specifies the host ID (IP address or domain name service (DNS) name) of the LDAP server.

Port:

Specifies the host port of the LDAP server.

If multiple application servers are installed and configured to run in the same single sign-on domain or if the application server interoperates with a previous version, it is important that the port number match all configurations. For example, if the LDAP port is explicitly specified as `389` in a Version 6.1 configuration, and a WebSphere Application Server at Version 7.0 is going to interoperate with the Version 6.1 server, verify that port `389` is specified explicitly for the Version 7.0 server.

Default: 389
Type: Integer

Base distinguished name (DN):

Specifies the base distinguished name (DN) of the directory service, which indicates the starting point for LDAP searches of the directory service. In most cases, bind DN and bind password are needed. However, when anonymous bind can satisfy all of the required functions, bind DN and bind password are not needed.

For example, for a user with a DN of `cn=John Doe , ou=Rochester, o=IBM, c=US`, specify the Base DN as any of the following options: `ou=Rochester, o=IBM, c=US` or `o=IBM, c=US` or `c=US`. For authorization purposes, this field is case sensitive. This specification implies that if a token is received, for example, from another cell or Lotus Domino, the base DN in the server must match the base DN from the other cell or Lotus Domino server exactly.

Bind distinguished name (DN):

Specifies the DN for the application server to use when binding to the directory service.

If no name is specified, the application server binds anonymously. See the Base distinguished name (DN) field description for examples of distinguished names.

Bind password:

Specifies the password for the application server to use when binding to the directory service.

Advanced Lightweight Directory Access Protocol user registry settings:

Use this page to configure the advanced Lightweight Directory Access Protocol (LDAP) user registry settings when users and groups reside in an external LDAP directory.

To view this administrative page, complete the following steps:

1. Click **Security > Global security**.

2. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.
3. Under Additional properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.

Default values for all the user and group related filters are already completed in the appropriate fields. You can change these values depending on your requirements. These default values are based on the type of LDAP server that is selected in the Standalone LDAP registry settings panel. If this type changes, for example from Netscape to Secureway, the default filters automatically change. When the default filter values change, the LDAP server type changes to `Custom` to indicate that custom filters are used. When security is enabled and any of these properties change, go to the Global security panel and click **Apply** or **OK** to validate the changes.

Note: The initial profile creation configures WebSphere Application Server to use a federated repositories security registry option with the file-based registry. This security registry configuration can be changed to use other options, including the stand-alone LDAP registry. Instead of changing from the federated repositories option to the stand-alone LDAP registry option under the User account repository configuration, consider employing the federated repositories option, which provides for LDAP configuration. Federated repositories provide a wide range of capabilities, including the ability to have one or multiple user registries. It supports federating one or more LDAPs in addition to file-based and custom registries. It also has improved failover capabilities, and a robust set of member (user and group) management capabilities. Federated repositories is required when you are using the new member management capabilities in WebSphere Portal 6.1 and above, and Process Server 6.1 and above. The use of federated repositories is required for following LDAP referrals, which is a common requirement in some LDAP server environments (such as Microsoft Active Directory).

It is recommended that you migrate from stand-alone LDAP registries to federated repositories. If you move to WebSphere Portal 6.1 and above, and or WebSphere Process Server 6.1 and above, you should migrate to federated repositories prior to these upgrades. For more information about federated repositories and its capabilities, read the Federated repositories topic. For more information about how to migrate to federated repositories, read the Migrating a stand-alone LDAP repository to a federated repositories LDAP repository configuration topic.

User filter:

Specifies the LDAP user filter that searches the user registry for users.

This option is typically used for security role-to-user assignments and specifies the property by which to look up users in the directory service. For example, to look up users based on their user IDs, specify `(&(uid=%v)(objectclass=inetOrgPerson))`. For more information about this syntax, see the LDAP directory service documentation.

Data type: String

Group filter:

Specifies the LDAP group filter that searches the user registry for groups

This option is typically used for security role-to-group assignments and specifies the property by which to look up groups in the directory service. For more information about this syntax, see the LDAP directory service documentation.

Data type: String

User ID map:

Specifies the LDAP filter that maps the short name of a user to an LDAP entry.

Specifies the piece of information that represents users when users display. For example, to display entries of the `object class = inetOrgPerson` type by their IDs, specify `inetOrgPerson:uid`. This field takes multiple objectclass:property pairs delimited by a semicolon (;).

Data type: String

Group ID map:

Specifies the LDAP filter that maps the short name of a group to an LDAP entry.

Specifies the piece of information that represents groups when groups display. For example, to display groups by their names, specify `*:cn`. The asterisk (`*`) is a wildcard character that searches on any object class in this case. This field takes multiple `objectclass:property` pairs, delimited by a semicolon (;).

Data type: String

Group member ID map:

Specifies the LDAP filter that identifies user-to-group relationships.

For directory types SecureWay, and Domino, this field takes multiple objectclass:property pairs, delimited by a semicolon (;). In an objectclass:property pair, the object class value is the same object class that is defined in the group filter, and the property is the member attribute. If the object class value does not match the object class in the group filter, authorization might fail if groups are mapped to security roles. For more information about this syntax, see your LDAP directory service documentation.

For IBM Directory Server, Sun ONE, and Active Directory, this field takes multiple `group attribute:member attribute` pairs delimited by a semicolon (;). These pairs are used to find the group memberships of a user by enumerating all the group attributes that are possessed by a given user. For example, attribute pair `memberof:member` is used by Active Directory, and ibm-allGroup:member is used by IBM Directory Server. This field also specifies which property of an object class stores the list of members belonging to the group represented by the object class. For supported LDAP directory servers, see "Supported directory services".

Data type: String

Perform a nested group search:

Specifies a recursive nested group search.

Select this option if the Lightweight Directory Access Protocol (LDAP) server does not support recursive server-side group member searches and if recursive group member search is required. It is not recommended that you select this option to locate recursive group memberships for LDAP servers. Application server security leverages the recursive search functionality of the LDAP server to search a user's group memberships, including recursive group memberships. For example:

- IBM Directory Server is preconfigured by the application server security to recursively calculate a user's group memberships using the `ibm-allGroup` attribute.
- SunONE directory server is preconfigured to calculate nested group memberships using the `nsRole` attribute.

Data type: String

Kerberos user filter:

Specifies the Kerberos user filter value. This value can be modified when Kerberos is configured and is active as one of the preferred authentication mechanisms.

Data type: String

Certificate map mode:

Specifies whether to map X.509 certificates into an LDAP directory by EXACT_DN or CERTIFICATE_FILTER. Specify CERTIFICATE_FILTER to use the specified certificate filter for the mapping.

Data type: String

Certificate filter:

Specifies the filter certificate mapping property for the LDAP filter. The filter is used to map attributes in the client certificate to entries in the LDAP registry.

If more than one LDAP entry matches the filter specification at runtime, authentication fails because the result is an ambiguous match. The syntax or structure of this filter is: `(&(uid=${SubjectCN})(objectclass=inetOrgPerson))`. The left side of the filter specification is an LDAP attribute that depends on the schema that your LDAP server is configured to use. The right side of the filter specification is one of the public attributes in your client certificate. The right side must begin with a dollar sign ($) and open bracket ({) and end with a close bracket (}). You can use the following certificate attribute values on the right side of the filter specification. The case of the strings is important:

- `${UniqueKey}`
- `${PublicKey}`
- `${IssuerDN}`
- `${Issuer<xx>}`

  where *<xx>* is replaced by the characters that represent any valid component of the Issuer Distinguished Name. For example, you might use `${IssuerCN}` for the Issuer Common Name.
- `${NotAfter}`
- `${NotBefore}`
- `${SerialNumber}`
- `${SigAlgName}`
- `${SigAlgOID}`
- `${SigAlgParams}`
- `${SubjectDN}`
- `${Subject<xx>}`

  where *<xx>* is replaced by the characters that represent any valid component of the Subject Distinguished Name. For example, you might use `${SubjectCN}` for the Subject Common Name.
- `${Version}`

Data type: String

Configuring Lightweight Directory Access Protocol search filters:

Use this topic to configure the LDAP search filters. These steps are required to modify existing user and group filters for a particular LDAP directory type, and also to set up certificate filters to map certificates to entries in the LDAP server.

Before you begin

WebSphere Application Server uses Lightweight Directory Access Protocol (LDAP) filters to search and obtain information about users and groups from an LDAP directory server. A default set of filters is provided for each LDAP server that the product supports. You can modify these filters to fit your LDAP configuration. After the filters are modified and you click **OK** or **Apply** the directory type in the Standalone LDAP registry panel changes to `custom`, which indicates that custom filters are used. Also, you can develop filters to support any additional type of LDAP server. The effort to support additional LDAP directories is optional and other LDAP directory types are not supported. Complete the following steps in the administrative console.

Procedure

1. Click **Security > Global security**.
2. Under User account repository, select **Standalone LDAP registry** and click **Configure**.
3. Under Additional properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.
4. Modify the user filter, if necessary. The user filter is used for searching the registry for users and is typically used for the security role-to-user assignment. The filter is also used to authenticate a user with the attribute that is specified in the filter. The filter specifies the property that is used to look up users in the directory service.

   In the following example, the property that is assigned to `%v`, which is the short name of the user, must be a unique key. Two LDAP entries with the same object class cannot have the same short name. To look up users based on their user IDs (uid) and to use the inetOrgPerson object class, specify the following syntax:

   ```
   (&(uid=%v)(objectclass=inetOrgPerson)
   ```

   For more information about this syntax, see the “Using specific directory servers as the LDAP server” on page 1275 documentation.
5. Modify the Kerberos user filter, if necessary. The Kerberos user filter name is used for searching the registry for the Kerberos principal name. Specify the LDAP attribute that holds the Kerberos principal name.

   IBM Lotus Domino default krbuser filter:
   (&(krbPrincipalName=%v)(objectcategory=Person))

   IBM SecureWay Directory Server default krbuser filter:
   (&(krbPrincipalName=%v)(objectcategory=ePerson))

   Microsoft Active Directory default krbuser filter:
   (&(userprincipalname=%v)(objectcategory=user))

   Sun Java System Directory Server default krbuser filter:
   (&(krbPrincipalName=%v)(objectcategory=inetOrgPerson))

   Novell eDirectory default krbuser filter:
   (&(krbPrincipalName=%v)(objectcategory=Person))
6. Optional: If your using Federated Repositories, modify the Kerberos attribute name if necessary. The Kerberos attribute name is used for searching the registry for Kerberos principal. Specify the LDAP attribute that holds the Kerberos principal name.

   IBM Lotus Domino default krbuser filter:
   krbPrincipalName

IBM SecureWay Directory Server default krbuser filter:
krbPrincipalName

Microsoft Active Directory default krbuser filter:
userprincipalname

Sun Java System Directory Server default krbuser filter:
krbPrincipalName

Novell eDirectory default krbuser filter:
krbPrincipalName

7. Modify the group filter, if necessary. The group filter is used in searching the registry for groups and is typically used for the security role-to-group assignment. Also, the filter is used to specify the property by which to look up groups in the directory service.

   In the following example, the property that is assigned to `%v`, which is the short name of the group, must be a unique key. Two LDAP entries with the same object class cannot have the same short name. To look up groups based on their common names (CN) and to use either the groupOfNames object class or the groupOfUniqueNames object class, specify the following syntax:

   ```
   (&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))
   ```

   For more information about this syntax, see the “Using specific directory servers as the LDAP server” on page 1275 documentation.
8. Modify the user ID map, if necessary. This filter maps the short name of a user to an LDAP entry and specifies the piece of information that represents users when these users are displayed with their short names. For example, to display entries of `object class = inetOrgPerson` by their IDs, specify `inetOrgPerson:uid`. This field takes multiple objectclass:property pairs, delimited by a semicolon (;). To provide a consistent value for methods like the getCallerPrincipal method and the getUserPrincipal method, the short name that is obtained by using this filter is used. For example, the `CN=Bob Smith, ou=austin.ibm.com, o=IBM, c=US` user can log in using any attributes that are defined, for example, email address, social security number, and so on, but when these methods are called, the `bob` user ID is returned no matter how the user logs in.
9. Modify the group ID map filter, if necessary. This filter maps the short name of a group to an LDAP entry and specifies the piece of information that represents groups when groups display. For example, to display groups by their names, specify `*:cn`. The asterisk (*) is a wildcard character that searches on any object class in this case. This field takes multiple objectclass:property pairs, delimited by a semicolon (;).
10. Modify the group member ID map filter, if necessary. This filter identifies user-to-group memberships. For SecureWay, and Domino directory types, this field is used to query all the groups that match the specified object classes to see if the user is contained in the specified attribute. For example, to get all the users that belong to groups with the groupOfNames object class and the users that are contained in the member attributes, specify `groupOfNames:member`. This syntax, which is a property of an object class, stores the list of members that belong to the group that is represented by the object class. This field takes multiple objectclass:property pairs that are delimited by a semicolon (;). For more information about this syntax, see the “Using specific directory servers as the LDAP server” on page 1275.

    For the IBM Tivoli Directory Server, Sun ONE, and Active Directory, this field is used to query all users in a group with the information that is stored in the user object. For example, the memberof:member filter (for Active Directory) is used to get the memberof attribute of the user object to obtain all the groups to which the user belongs. The member attribute is used to get all the users in a group that use the Group object. Using the User object to obtain the group information improves performance.
11. Select the **Perform a nested group search** option if your LDAP server does not support recursive server-side searches.
12. Modify the Certificate map mode, if necessary. You can use the X.590 certificates for user authentication when LDAP is selected as the registry. This field is used to indicate whether to map

the X.509 certificates into an LDAP directory user by **EXACT_DN** or **CERTIFICATE_FILTER**. If **EXACT_DN** is selected, the DN in the certificate must exactly match the user entry in the LDAP server, including case and spaces.

Select the **Ignore case for authorization** option on the Standalone LDAP registry settings to make the authorization case insensitive. To access the Standalone LDAP registry settings panel, complete the following steps:

a. Click **Security > Global security**.

b. Under User account repository, click the **Available realm definitions** drop-down list, select**Standalone LDAP registry**.

13. If you select **CERTIFICATE_FILTER**, specify the LDAP filter for mapping attributes in the client certificate to entries in LDAP. If more than one LDAP entry matches the filter specification at run time, authentication fails because an ambiguous match results. The syntax or structure of this filter is: `LDAP attribute=${Client certificate attribute}` (for example, `uid=${SubjectCN}`).

    The left side of the filter specification is an LDAP attribute that depends on the schema that your LDAP server is configured to use. The right side of the filter specification is one of the public attributes in your client certificate. Note that the right side must begin with a dollar sign ($), open bracket ({), and end with a close bracket (}). Use the following certificate attribute values on the right side of the filter specification. The case of the strings is important.

    - `${UniqueKey}`
    - `${PublicKey}`
    - `${IssuerDN}`
    - `${Issuer<xx>}`

      where *<xx>* is replaced by the characters that represent any valid component of the Issuer Distinguished Name. For example, you might use `${IssuerCN}` for the Issuer Common Name.
    - `${NotAfter}`
    - `${NotBefore}`
    - `${SerialNumber}`
    - `${SigAlgName}`
    - `${SigAlgOID}`
    - `${SigAlgParams}`
    - `${SubjectDN}`
    - `${Subject<xx>}`

      where *<xx>* is replaced by the characters that represent any valid component of the Subject Distinguished Name. For example, you might use `${SubjectCN}` for the Subject Common Name.
    - `${Version}`

    To enable this field, select **CERTIFICATE_FILTER** for the certificate mapping.

14. Click **Apply**.

    When any LDAP user or group filter is modified in the Advanced LDAP Settings panel click **Apply**. Clicking **OK** navigates you to the Standalone LDAP registry panel, which contains the previous LDAP directory type, rather than the custom LDAP directory type. Clicking **OK** or **Apply** in the Standalone LDAP registry panel saves the back-level LDAP directory type and the default filters of that directory. This action overwrites any changes to the filters that you made. To avoid overwriting changes, you can take either of the following actions:

    - Click **Apply** in the Advanced Lightweight Directory Access Protocol (LDAP) user registry settings panel. Click **Security > Global security** and change the User account repository type to `Stand-alone custom registry`.
    - Select **Custom** type from the Standalone LDAP registry panel. Click **Apply** and then change the filters by clicking the Advanced Lightweight Directory Access Protocol (LDAP) user registry settings panel. After you complete your changes, click **Apply** or **OK**.

    The validation of the changes does not take place in this panel. Validation is done when you click **OK** or **Apply** on the Global security panel. If you are in the process of enabling security for the first time, complete the remaining steps and go to the Global security panel. Select **Standalone LDAP registry** as the user account repository. If security is already enabled and any information on this panel

changes, go to the Global security panel and click **OK** or **Apply** to validate your changes. If your changes are not validated, the server might not start.

Results

These steps result in the configuration of the LDAP search filters. These steps are required to modify existing user and group filters for a particular LDAP directory type. The steps are also used to set up certificate filters to map certificates to entries in the LDAP server.

What to do next

1. Validate this setup by clicking **OK** or **Apply** on the Global security panel.
2. Save, stop, and start all the product servers, including the cell, nodes and all of the application servers for any changes in this panel to become effective.
3. After the server starts, go through all the security-related tasks (getting users, getting groups, and so on) to verify that the changes to the filters function.

Using specific directory servers as the LDAP server:

This article provides important information about the directory servers that are supported as Lightweight Directory Access Protocol (LDAP) servers in WebSphere Application Server.

Before you begin

Microsoft Active Directory forests are not supported with the stand-alone LDAP Registry. The Federated Repository Registry, when configured to use an Active Directory LDAP does support the use of forests.

About this task

For a list of supported LDAP servers, refer to the Supported hardware and software website.

It is expected that other LDAP servers follow the LDAP specification. Support is limited to these specific directory servers only. You can use any other directory server by using the custom directory type in the list and by filling in the filters that are required for that directory.

To improve performance for LDAP searches, the default filters for IBM Tivoli Directory Server, Sun ONE, and Active Directory are defined such that when you search for a user, the result contains all the relevant information about the user (user ID, groups, and so on). As a result, the product does not call the LDAP server multiple times. This definition is possible only in these directory types, which support searches where the complete user information is obtained.

If you use the IBM Directory Server, select the **Ignore case for authorization** option. This option is required because when the group information is obtained from the user object attributes, the case is not the same as when you get the group information directly. For the authorization to work in this case, perform a case insensitive check and verify the requirement for the **Ignore case for authorization** option.

- **Using Directory Services as the LDAP server**

  Support for groups that contain other groups or nested groups depends upon the specific versions of WebSphere Application Server and LDAP. For more information, see “Dynamic groups and nested group support for LDAP” on page 1408.
- **Using IBM Tivoli Directory Server as the LDAP server**

  To use IBM Tivoli Directory Server, formerly IBM Directory Server, select **IBM Tivoli Directory Server** as the directory type.

  The difference between these two types is group membership lookup. It is recommended that you choose the IBM Tivoli Directory Server for optimum performance during runtime. In the IBM Tivoli Directory Server, the group membership is an operational attribute. With this attribute, a group

membership lookup is done by enumerating the ibm-allGroups attribute for the entry. All group memberships, including the static groups, dynamic groups, and nested groups, can be returned with the `ibm-allGroups` attribute.

WebSphere Application Server supports dynamic groups, nested groups, and static groups in IBM Tivoli Directory Server using the `ibm-allGroups` attribute. To utilize this attribute in a security authorization application, use a case-insensitive match so that attribute values returned by the ibm-allGroups attribute are all in uppercase.

Important: It is recommended that you do not install IBM Tivoli Directory Server Version 6.0 on the same machine that you install Version 8.0. IBM Tivoli Directory Server Version 6.0 includes WebSphere Application Server, Express Version 5.1.1, which the directory server uses for its administrative console. Install the Web Administration tool Version 6.0 and WebSphere Application Server, ExpressVersion 5.1.1, which are both bundled with IBM Tivoli Directory Server Version 6.0, on a different machine from Version 8.0. You cannot use Version 8.0 as the administrative console for IBM Tivoli Directory Server. If IBM Tivoli Directory Server Version 6.0 and Version 8.0 are installed on the same machine, you might encounter port conflicts.

If you must install IBM Tivoli Directory Server Version 6.0 and Version 8.0 on the same machine, consider the following information:

- During the IBM Tivoli Directory Server installation process, you must select both the **Web Administration tool** and **WebSphere Application Server, Express Version 5.1.1**.
- Install Version 8.0.
- When you install Version 8.0, change the port number for the application server.
- You might need to adjust the WebSphere Application Server environment variables on Version 8.0 for *WAS_HOME* and *WAS_INSTALL_ROOT* (or *APP_SERVER_ROOT* for IBM i). To change the variables using the administrative console, click **Environment > WebSphere Variables**.

- **Using a Lotus Domino Enterprise Server as the LDAP server**

  If you select the Lotus Domino Enterprise Server Version 6.5.4 or Version 7.0 and the attribute short name is not defined in the schema, you can take either of the following actions:
  - Change the schema to add the short name attribute.
  - Change the user ID map filter to replace the short name with any other defined attribute (preferably to UID). For example, change `person:shortname` to `person:uid`.

  The userID map filter is changed to use the uid attribute instead of the shortname attribute as the current version of Lotus Domino does not create the shortname attribute by default. If you want to use the shortname attribute, define the attribute in the schema and change the userID map filter.

  ```
  User ID Map :    person:shortname
  ```

- **Using Sun ONE Directory Server as the LDAP server**

  You can select **Sun ONE Directory Server** for your Sun ONE Directory Server system. In Sun ONE Directory Server, the object class is the default groupOfUniqueName when you create a group. For better performance, WebSphere Application Server uses the User object to locate the user group membership from the nsRole attribute. Create the group from the role. If you want to use the groupOfUniqueName attribute to search groups, specify your own filter setting. Roles unify entries. Roles are designed to be more efficient and easier to use for applications. For example, an application can locate the role of an entry by enumerating all the roles that are possessed by a given entry, rather than selecting a group and browsing through the members list. When using roles, you can create a group using a:
  - Managed role
  - Filtered role
  - Nested role

All of these roles are computable by the nsRole attribute.

- **Using Microsoft Active Directory server as the LDAP server**

  To use Microsoft Active Directory as the LDAP server for authentication with WebSphere Application Server you must take specific steps. By default, Microsoft Active Directory does not permit anonymous LDAP queries. To create LDAP queries or to browse the directory, an LDAP client must bind to the LDAP server using the distinguished name (DN) of an account that has the authority to search and read the values of LDAP attributes, such as user and group information, needed by the Application Server. A group membership search in the Active Directory is done by enumerating the `memberof` attribute for a given user entry, rather than browsing through the member list in each group. If you change the default behavior to browse each group, you can change the **Group Member ID Map** field from `memberof:member` to `group:member`.

The following steps describe how to set up Microsoft Active Directory as your LDAP server.

Procedure

1. Determine the full distinguished name (DN) and password of an account in the administrators group.

   For example, if the Active Directory administrator creates an account in the Users folder of the Active Directory Users and Computers Windows control panel and the DNS domain is `ibm.com`, the resulting DN has the following structure:

```
cn=<adminUsername>, cn=users, dc=ibm,
dc=com
```

2. Determine the short name and password of any account in the Microsoft Active Directory.
3. Use the WebSphere Application Server administrative console to set up the information that is needed to use Microsoft Active Directory.
   a. Click **Security > Global security**.
   b. Under User account repository, select **Standalone LDAP registry** and click **Configure**.
   c. Set up LDAP with Active Directory as the type of LDAP server. Based on the information that is determined in the previous steps, you can specify the following values on the LDAP settings panel:

      Primary administrative user name
      Specify the name of a user with administrative privileges that is defined in the registry. This user name is used to access the administrative console or used by wsadmin.

      Type Specify Active Directory

      Host Specify the domain name service (DNS) name of the machine that is running Microsoft Active Directory.

      Base distinguished name (DN)
      Specify the domain components of the DN of the account that is chosen in the first step. For example: `dc=ibm, dc=com`

      Bind distinguished name (DN)
      Specify the full distinguished name of the account that is chosen in the first step. For example: `cn=adminUsername, cn=users, dc=ibm, dc=com`

      Bind password
      Specify the password of the account that is chosen in the first step.

   d. Click **OK** and **Save** to save the changes to the master configuration.
4. Click **Security > Global security**.
5. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.
6. Select either the **Automatically generated server identity** or **Server identity that is stored in the repository** option. If you select the **Server identity that is stored in the repository** option, enter the following information:

Server user ID or administrative user on a Version 6.0.x node
Specify the short name of the account that is chosen in the second step.

Server user password
Specify the password of the account that is chosen in the second step.

7. Optional: Set ObjectCategory as the filter in the Group member ID map field to improve LDAP performance.
   a. Under Additional properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings** .
   b. Add `;objectCategory:group` to the end of the Group member ID map field.
8. Click **OK** and **Save** to save the changes to the master configuration.
9. Stop and restart the administrative server so that the changes take effect.

Adding users to the Lightweight Directory Access Protocol user registry:

You can use the Lightweight Directory Access Protocol (LDAP) user registry with any of the authentication mechanisms supported by WebSphere Application Server. Therefore, it is necessary to add users into the LDAP directory that you want to have authorization to access Application Server resources.

About this task

This information in this article is specific to the iSeries Directory Services product.

A variety of methods are available to add users. However, the easiest way is to create an LDAP Data Interchange Format (LDIF) file. The file contains the set of users to add into the directory. The file is used by the LDAP utilities, such as **idsldapmodify**. You can run these utilities from either the operating system or from a workstation. If you run these LDAP utilities from the operating system, your LDIF file must reside in the integrated file system.

Complete the following steps to add users to the LDAP user registry:

Procedure

1. Create an LDIF file and save it in the integrated file system. Use either the Edit File (EDTF) utility or your workstation text editor to create the file. Save the file in the integrated file system either by mapping a drive or using the file transfer protocol (FTP).

   For WebSphere Application Server and LDAP directory services, create entries in the directory that correspond to the ePerson schema definition.

   A simple ePerson LDIF entry resembles the following example:

```
dn: cn=John Doe, ou=Rochester, o=IBM, c=US
objectclass: person
objectclass: inetOrgPerson
objectclass: top
objectclass: organizationalPerson
objectclass: ePerson
cn: John Doe
sn: Doe
uid: jdoe
userpassword: secretpass
```

   This LDIF entry defines an ePerson for user John Doe. The user identification (uid) for John is set to `jdoe` and his password is set to `secretpass`. This entry resides within the Rochester organizational unit, which is within the IBM organization in the United States. Each of the `ou`, `o`, and `c` containing entries are defined before this ePerson entry is defined. You can define a series of LDIF entries in the same file to define Lightweight Third Party Authentication (LTPA) users for WebSphere Application Server.

   If you do not specify a value for the userpassword attribute, the LDAP server attempts to authenticate LTPA users with the user profile for the local operating system that is identified by the uid attribute value. This action might be desirable if users have user profiles for the operating system and do not want to manage passwords in both the operating system user registry and the LDAP directory.

When you create an ePerson entry, make sure that the `cn` and `uid` attributes each have a unique value. Do not create two entries that have the same value for the `cn` and `uid` attributes.

Important: If you have a large user registry, login performance might be severely impacted if the Group Member ID Map property is left at its default value, which is both groupOfNames:member and groupOfUniqueNames:uniqueMember.

To address this performance problem, specify one of these object classes and not both. You must then exclusively use the selected object class to implement groups in the user registry.

2. Import the LDIF file entries into your directory on the server. Use the LDAP ldapadd utility in Qshell Interpreter (QSH) or from a workstation.

What to do next

For more information on importing LDIF entries, see the Directory Services documentation in the Information Centers for IBM i 6.1 and 7.1.

Locating user group memberships in a Lightweight Directory Access Protocol registry:

You can configure WebSphere Application Server security to use Lightweight Directory Access Protocol (LDAP) servers. The LDAP specifications allow for different mechanisms to define group memberships. Depending on your LDAP server implementation, you can use methods to determine group memberships. WebSphere Application Server can search group memberships directly or indirectly. Also, you can configure the product to search one or more static groups, recursive or nested groups, and dynamic groups for some Lightweight Directory Access Protocol (LDAP) servers.

Procedure

- Evaluate group memberships.
  - **Static group membership:** All LDAP server implementations support static group membership. The group object contains a list of users or groups that are members of the group. To determine the groups in which a user is a member, you must get the list of all groups, and then query each group in turn to see if the user is a member of that group. This operation results in (0)zero groups and does not scale well.

    Several LDAP servers enable user objects in the LDAP server to contain information about the groups to which they belong. Examples of LDAP servers that support direct group searches include Microsoft Active Directory Server and the owner of eDirectory.
  - **Dynamic group memberships**

    Some user group memberships are computable from attributes within the user object. IBM Directory Server and Sun ONE Directory Server are two examples of LDAP servers that support dynamic group membership. In some LDAP servers, you can use an attribute to include a user's dynamic group memberships, nesting group memberships, and static group memberships to determine all the group memberships from a single attribute.

    For example, in IBM Directory Server, you can return all group memberships including the static groups, dynamic groups, and nested groups using the ibm-allGroups attribute. In the Sun ONE directory server you can use the nsRole attribute to calculate, all roles, including managed roles, filtered roles, and nested roles. If an LDAP server has such an attribute in a User object to include dynamic groups, nested groups, and static groups, you can configure WebSphere Application Server security to use this attribute to determine these groups.

    Depending on the implementation, and LDAP server can caluculate dynamic group membership. In this case, this dynamic computation is performed entirely by the LDAP server under a single LDAP query and is invisible to WebSphere Application Server. While this approach is not as efficient as direct groups, server-side dynamic queries are more efficient than determining group membership using static group queries.

    Dynamic group membership, when it is supported by the LDAP server, is frequently reflected back to the LDAP client, which is the WebSphere Application Server. In this configuration, WebSphere

Application Server is required to compose the appropriate dynamic query against LDAP for each group. This operation results in 0(zero) groups and does not scale well.

Tips:

– Use the efficient direct group membership where possible.

– Use the relatively efficient dynamic group membership where the LDAP computes membership within a single query.

– Use static group membership, or client side dynamic group membership as a secondary alternative. This option only performs well on systems where the number of groups within the LDAP server is "small".

The configurations for the supported, listed LDAP servers are pre-defined to use the optimal group membership mechanisms. They assume that the standard object types and schemas for that LDAP vendor are in use on the LDAP server.

• Evaluate the LDAP registry configuration.

– **Standalone LDAP registry**

If you are configuring an LDAP server outside of the list of pre-configured types, you must configure the appropriate value in the Group Member ID map field on the Advanced LDAP Settings panel using the following methods.

- If you use static group membership, you must specifiy objectclass:attribute pairs. If the objectclass for the group object is, **groupOfUniquePersons**, and within that objectclass, members are listed as **persons**, then the static group membership **Group Member ID map** is **groupOfUniquePersons:persons**.
- If direct group membership is used, attributes exist in the objectclass, you must use attribute:attribute pairs. For example, if the objectclass for the user is **user**and the objectclasst contains attributes called **ingroup**, which contains each group membership, then the direct group membership **Group Member ID map** is **ingroup:member**.

– **LDAP Registry within a Federated Repositories Registry**

If you are configuring an LDAP server outside of the list of pre-configured types, you must configure the appropriate value in the **Group attribute definition** properties for the repository.

- If static group membership is used, you must specify the name of the object class, and the attribute that is used for indicating membership in **Group attribute definition -> Member attributes**. If the group objectclass for the user is, **groupOfUniquePersons**, and within that objectclass, members are listed as **persons**, then the static group **Member attributes** property is set follows:
  1. In the administrative console, click **Security > Global security**.
  2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
  3. Under Related items, click **Manage repositories**.
  4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
  5. Under Additional properties, click **Group attribute definition**.
  6. Under Additional properties, click **Member attributes**.
  7. Click **New** to specify a new member attribute.

  Set the **Name of member attribute** field to `persons`

  Set the **Object class field** to `groupOfUniquePersons`

  When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.
- If direct group membership is used, then attributes exist in the objectclass for the user and you must use the attribute. For example, if the objectclass for the user is **user**, and it contains

attributes called **ingroup** that contain each group membership, then you specify the direct group membership in the **Group attribute definition** property for the repository. Perform the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **Group attribute definition**.

Set the **Name of group membership attribute** field to `ingroup`.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

- Use the sample entries of objectClass:attribute pairs in the Group member ID map field.
  - dominoGroup:member for Lotus Domino
  - groupOfNames:member for eDirectory
- Evaluate Nested Groups.
  - **Nested Groups**

    Depending on the LDAP server implementation, groups can contain only users, or can contain other groups, which is known as a nested group. You configure WebSphere Application Server to properly discover all groups by following this nesting as it applies to either a stand-alone LDAP registry or a LDAP Registry within a Federated Repositories Registry.

    - **Standalone LDAP Registry** The stand-alone LDAP registry default setting performs only a single group membership query. If the groups returned are in fact subgroups of other groups, you must enable the **Perform a nested group search** property on the Advanced LDAP Settings panel of the LDAP registry as follows:
      1. Click **Security > Global security**.
      2. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.
      3. Under Additional properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.

      Put a check mark in the **Perform a nested group search** check box.
    - **LDAP Registry within a Federated Repositories Registry** Within Federated repositories, you must configure what you expect the results of the query to return. Based on this information, the Federated repository makes the appropriate calls to establish all group membership. If the LDAP server returns all nested group information within a single direct group query, then you set the **Scope of group membership** attribute property in the group attribute definition to **Nested.** as follows:
      1. In the administrative console, click **Security > Global security**.
      2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
      3. Under Related items, click **Manage repositories**.
      4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
      5. Under Additional properties, click **Group attribute definition**.

      Set the **Scope of group membership** attribute property in the group attribute definition to **Nested.**

- If the LDAP server returns only the direct membership, then the registry must then make subsequent queries to establish complete membership. To force the Federated Repository to issue subsequent queries, set the **Scope of group membership** attribute property in the **Group attribute definition** for the repository to **Direct**.

Results

While using the direct method, dynamic groups, recursive groups, and static groups can be returned as multiple values of a single attribute. For example, in IBM Directory Server all group memberships, including the static groups, dynamic groups, and nested groups, can be returned using the ibm-allGroups attribute. In Sun ONE, all roles, including managed roles, filtered roles, and nested roles, are calculated using the nsRole attribute. If an LDAP server can use the nsRole attribute, dynamic groups, nested groups, and static groups are all supported by WebSphere Application Server.

Some LDAP servers do not have recursive computing functionality. For example, although Microsoft Active Directory server has direct group search capability using the memberOf attribute, this attribute lists the groups beneath, which the group is directly nested only and does not contain the recursive list of nested predecessors. The Lotus Domino LDAP server only supports the indirect method to locate the group memberships for a user. You cannot obtain recursive group memberships from a Domino server directly. For LDAP servers without recursive searching capability, WebSphere Application Server security provides a recursive function that is enabled by clicking **Perform a Nested Group Search** in the Advanced LDAP user registry settings. Select this option only if your LDAP server does not provide recursive searches and you want a recursive search.

Configuring dynamic and nested group support for the SunONE or iPlanet Directory Server:

Configure dynamic and nested groups to simplify WebSphere Application Server security management and increase its effectiveness and flexibility.

Before you begin

To use dynamic and nested groups with WebSphere Application Server security, you must be running WebSphere Application Server Version 6.1 or later. Refer to “Dynamic groups and nested group support for LDAP” on page 1408 for more information on this topic.

Procedure

1. In the administrative console for WebSphere Application Server, click **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.
3. Select `SunONE` for the type of LDAP server.
4. Select the **Ignore case for authorization** option.
5. Under Additional Properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.
6. Change the Group filter setting to `&(cn=%v)(objectclass=ldapsubentry))`.
7. Change the Group member ID map setting to `nsRole:nsRole`.
8. Click **Apply** or **OK** to validate the changes.

Configuring dynamic and nested group support for the IBM Tivoli Directory Server:

Configure dynamic and nested groups to simplify WebSphere Application Server security management and increase its effectiveness and flexibility.

Before you begin

When creating groups, ensure that nested and dynamic group memberships work correctly.

Procedure

1. In the administrative console for WebSphere Application Server, click **Security > Global security**.
2. Under User account repository, click **Standalone LDAP registry**, and click **Configure**.
3. Select `IBM Tivoli Directory Server` for the type of LDAP server.
4. Under Additional properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.
5. Change the Group filter value to `(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))`.
6. Change the Group member ID map value to `ibm-allGroups:member;ibm-allGroups:uniqueMember`.
7. Click **Apply** or **OK** to validate the changes.
8. Verify that Auxiliary object class field on the Add an LDAP entry panel for your IBM Tivoli Directory server has the appropriate value. When you create a nested group, the Auxiliary object class value is `ibm-nestedGroup`. When you create a dynamic group, the Auxiliary object class value is `ibm-dynamicGroup`.

Configuring multiple LDAP servers for user registry failover:

WebSphere Application Server security can be configured to attempt failovers between multiple Lightweight Directory Access Protocol (LDAP) hosts.

Before you begin

The multiple LDAP servers involved in the failover can be replicas that are replicated from the same master LDAP server, or they can be any LDAP host with the same schema. That is any LDAP host that contains data that is imported from the same LDAP data interchange format (LDIF) file.

Note: When WebSphere Application Server attempts failovers between multiple Lightweight Directory Access Protocol (LDAP) hosts, system properties are exchanged. WebSphere Application Server Version 6.1.0 manages the SSL configuration and these system properties. You cannot expect to set system properties yourself and expect the failover to succeed.

Procedure

1. Start the deployment manager process.
   a. Start the Command Prompt application.
   b. Change directories to *profile_root*`/bin`.
   c. Enter startManager.
2. Start the wsadmin Command Prompt application.
   a. Start the Command Prompt application.
   b. Change directories to *profile_root*`/bin`.
   c. Enter the following command:
      `wsadmin –user` *username* `–password` *password*
3. Configure a second LDAP server for failover.
   a. Enter the following command to set the failover LDAP server hostname:
      `set ldapServer [`*ldap server hostname*`]`
   b. Enter the following command to set the LDAP server port number:
      `set ldapPort [`*ldap server port*`]`
   c. Enter the following command to set the WebSphere LDAP failover variable:

```
set Attrs2 [list [list hosts [list [list [list host $ldapServer] [list port $ldapPort]]]]]
```

d. Modify the LDAP configuration to add the failover LDAP server by entering the following command:

```
set result [$AdminConfig list LDAPUserRegistry]
```

e. Find the LDAP server configID by entering the following command:

```
$AdminConfig modify $result $Attrs2
```

f. Enter the following command to save the configuration change:

```
$AdminConfig save
```

g. Enter exit to quit the Command Prompt application. The following is an example of the Command Prompt application output:

```
wsadmin>set ldapServer [list xxxx.xxxx.xxx.com]
xxxx.xxxx.xxx.com
wsadmin>set ldapPort [list NNN]
NNN
wsadmin>set Attrs2 [list [list hosts [list [list [list host $ldapServer] [list port $ldapPort]]]]]
{hosts {{{host xxxx.xxxx.xxx.com} {port NNN}}}}
wsadmin> set result [$AdminConfig list LDAPUserRegistry]
(cells/Father2Cell01|security.xml#LDAPUserRegistry_1)
wsadmin>$AdminConfig modify $result $Attrs2

wsadmin>$AdminConfig save
```

4. Review the configuration change by opening the `security.xml` file with a text editor and review the new entry.
5. Stop the deployment manager.
   a. Start the Command Prompt application.
   b. Change directories to *profile_root*/`bin`.
   c. To stop the deployment manager, enter the following command:

```
stopManager –user username –password password
```

Testing an LDAP server for user registry failover:

After configuring a Lightweight Directory Access Protocol (LDAP) host for failover you should test the failover server by stopping the main LDAP server.

Before you begin

This task assumes the following setup:

- Deployment Manager is installed on the primary LDAP server running Application Server version 6.0.2 or higher.
- All other LDAP hosts are Active Directory machines with similar user registry designs.
- Atleast one of the other LDAP hosts has been configured for failover.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Procedure

1. Stop the Active Directory Server on the failover server.
2. Start the deployment manager process.
   a. Start the Command Prompt application.

b. Change directories to *profile_root*/bin.
c. Enter startManager.

3. Review the SystemOut.log file to see if the LDAP failover happened. The sample text is an example of a SystemOut.log file that records a successful failover:

```
[7/11/05 15:38:31:324 EDT] 0000000a LdapRegistryI A   SECJ0418I:
Cannot connect to the LDAP server ldap://xxxx.xxxxx.xxxx.com:NNN. {primary LDAP server}
[7/11/05 15:38:32:486 EDT] 0000000a UserRegistryI A   SECJ0136I:
Custom Registry:com.ibm.ws.security.registry.ldap.LdapRegistryImpl has been initialized
[7/11/05 15:38:53:787 EDT] 0000000a LdapRegistryI A   SECJ0419I:
The user registry is currently connected to the LDAP server ldap://xxxx.xxxxx.xxxx.com:NNN. {failover LDAP server}
...
[7/11/05 15:39:35:667 EDT] 0000000a WsServerImpl  A   WSVR0001I: Server dmgr open for e-business
```

4. Log into the console to see working and non-working cases.
   a. Start a browser.
   b. Browse to http://localhost:9060/admin.
   c. Type in your user ID and password and click **OK**.
   d. Log out of the Administrative Console.
   e. Type in DummyAdmin as the user ID and dummy1admin as your password and click **OK**. This should fail proving WebSphere Application Server is connected to the other LDAP server. Please make sure that on a production system the user registries are identical so this problem does not happen when switching between LDAP servers.
5. Stop the deployment manager.
   a. Start the Command Prompt application.
   b. Change directories to *profile_root*/bin.
   c. To stop the deployment manager, enter the following command:

```
stopManager –user username –password password
```

Deleting LDAP endpoints using wsadmin:

You can delete Lightweight Directory Access Protocol (LDAP) endpoints for a user registry by using the WebSphere Application Server administrative tool (wsadmin).

Procedure

1. Start the wsadmin scripting tool.
2. Set the LDAP variable and display a list of LDAP endpoint objects. Enter the following commands:

   Using Jacl:

```
set ldap [$AdminConfig list LDAPUserRegistry]

$AdminConfig list EndPoint $ldap
```

   Using Jython:

```
ldap=AdminConfig.list["LDAPUserRegistry"]

print AdminConfig.show(ldap)
```

   For the Jython language, you can obtain the endpoint from the host variable after running the previous command.
3. Display a list of LDAP endpoint objects. Enter the following command for each object:

   Using Jacl:

```
$AdminConfig showall End_Point_Object
```

   Using Jython:

```
AdminConfig.showall("End_Point_Object")
```

4. Delete an LDAP endpoint object. Enter the following command:

Using Jacl:

```
$AdminConfig remove End_Point_Object
```

Using Jython:

```
AdminConfig.remove ("End_Point_Object")
```

5. Save your configuration changes: Enter the following command:

   Using Jacl:

   ```
   $AdminConfig save
   ```

   Using Jython:

   ```
   AdminConfig.save()
   ```

Updating LDAP binding information:

Use this information to dynamically update security LDAP binding information by switching to a different binding identity.

About this task

You can dynamically update Lightweight Directory Access Protocol (LDAP) binding information without first stopping and restarting WebSphere Application Server by using the **wsadmin** tool.

The resetLdapBindInfo method in SecurityAdmin MBean is used to dynamically update LDAP binding information at WebSphere Application Server security runtime, and it takes the bind distinguished name (DN) and bind password parameters as input. The resetLdapBindInfo method validates the bind information against the LDAP server. If validation passes, new binding information is stored in `security.xml`, and a copy of the information is placed in WebSphere Application Server security runtime.

If the new binding information is `null`, `null`, the resetLdapBindInfo method first extracts LDAP binding information, including bind DN, bind password, and target binding host from WebSphere Application Server security configuration in `security.xml`. It then pushes the binding information to WebSphere Application Server security runtime.

There are two ways to dynamically update WebSphere Application Server security LDAP binding information using the SecurityAdmin MBean through wsadmin:

- "Switching to a different binding identity"
- "Switching to a failover LDAP host" on page 1287

Switching to a different binding identity:
About this task

To dynamically update security LDAP binding information by switching to a different binding identity:

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.
3. Create a new bind DN. It must have the same access authority as the current bind DN.
4. Run the SecurityAdmin MBean across all of the application server processes to validate the new binding information, to save it to `security.xml`, and to push the new binding information to the runtime.

Example

The following is a sample Jacl file for step 4:

```
proc LDAPReBind {args} {
  global AdminConfig AdminControl ldapBindDn ldapBindPassword
  set ldapBindDn [lindex $args 0]
  set ldapBindPassword [lindex $args 1]
       set secMBeans [$AdminControl queryNames type=SecurityAdmin,*]
       set plist  [list $ldapBindDn $ldapBindPassword]
       foreach secMBean $secMBeans {
            set result [$AdminControl invoke $secMBean resetLdapBindInfo $plist]
       }
 }
```

Switching to a failover LDAP host:
About this task

To dynamically update security LDAP binding information by switching to a failover LDAP host:

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Standalone LDAP registry** and click **Configure**.
3. Change the password for bind DN on one LDAP server (it can be the primary or the backup).
4. Update the new bind DN password to WebSphere Application security runtime by calling resetLdapBindInfo with the bind DN and by using its new password as a parameter.
5. Use the new bind DN password for all of the other LDAP servers. The binding information is now consistent across WebSphere Application Server and the LDAP servers.

   If you call resetLdapBindInfo with `null`, `null` as input parameters, WebSphere Application Server security runtime completes the following steps:

   a. Reads the bind DN, bind password, and target LDAP hosts from `security.xml`.

   b. Refreshes the cached connection to the LDAP server.

   If you configure security to use multiple LDAP servers, this MBean call forces WebSphere Application Server security to reconnect to the first available LDAP host in the list. For example, if three LDAP servers are configured in the order of L1, L2, and L3, the reconnection process always starts with the L1 server.

   When LDAP failover is configured by associating a single hostname to multiple IP addresses, entering an invalid password can cause multiple LDAP bind retries. With the default settings, the number of LDAP bind retries is equal to one more than the number of associated IP addresses. This means a single invalid login attempt can cause the LDAP account to be locked. If the com.ibm.websphere.security.ldap.retryBind custom property is set to false, LDAP bind calls are not retried.

   gotcha: Federated repository does not support failover by associating a single hostname to multiple IP addresses. This feature is only available in stand-alone LDAP.

Configuring stand-alone custom registries

Use the following information to configure stand-alone custom registries through the administrative console.

Before you begin

Before you begin this task, implement and build the UserRegistry interface. For more information on developing stand-alone custom registries refer to Developing stand-alone custom registries. The following steps are required to configure stand-alone custom registries through the administrative console.

Procedure

1. Click **Security > Global security**.
2. Under User account repositories, select **Stand-alone custom registry** and click **Configure**.
3. Enter a valid user name in the Primary administrative user name field. This ID is the security server ID, which is only used for WebSphere Application Server security and is not associated with the

system process that runs the server. The server calls the local operating system registry to authenticate and obtain privilege information about users by calling the native APIs in that particular registry.

4. Enter the dot-separated class name that implements the com.ibm.websphere.security.UserRegistry interface in the Custom registry class name field. For the sample, this file name is `com.ibm.websphere.security.FileRegistrySample`.

   Attention: The sample provided is intended to familiarize you with this feature. Do not use this sample in an actual production environment.

5. Add your custom registry class name to the class path.
6. Optional: Select the **Ignore case for authorization** option for the authorization to perform a case insensitive check. Enabling this option is necessary only when your user registry is case insensitive and does not provide a consistent case when queried for users and groups.
7. Click **Apply** if you have any other additional properties to enter for the registry initialization.
8. Optional: Enter additional properties to initialize your implementation.
   a. Click **Custom properties > New**.
   b. Enter the property name and value.

      For the sample, enter the following two properties. It is assumed that the `users.props` file and the `groups.props` file are in the *customer_sample* directory under the product installation directory. You can place these properties in any directory that you choose and reference their locations through custom properties. However, make sure that the directory has the appropriate access permissions.

Table 82. Additional properties.

This table lists additional custom properties when configuring stand-alone custom registries.

Property name	Property value
usersFile	${*USER_INSTALL_ROOT*}/*customer_sample* /users.props
groupsFile	${*USER_INSTALL_ROOT*}/*customer_sample* /groups.props

      Attention: The QEJBSVR user profile must have Execute (*X) authority for the directory that contains `user.props` and `groups.props` files. Additionally, QEJBSVR must have Read and Execute (*RX) authority for the `user.props` and `groups.props` files.

      Samples of these two properties are available in “users.props file” on page 1310 and “groups.props file” on page 1310.

      The **Description**, **Required**, and **Validation Expression** fields are not used and can remain blank.

      WebSphere Application Server version 4-based custom user registry is migrated to the custom user registry based on the com.ibm.websphere.security.UserRegistry interface.

   c. Click **Apply**.
   d. Repeat this step to add other additional properties.
9. Click **Security > Global security**.
10. Under User account repository, click the **Available realm definitions** drop-down list, select **Stand-alone custom registry**, and click **Configure**.
11. Select either the **Automatically generated server identity** or **Server identity that is stored in the repository** option. If you select the **Server identity that is stored in the repository** option, enter the following information:

    Server user ID or administrative user on a Version 6.0.x node
    Specify the short name of the account that is chosen in the second step.

    Server user password
    Specify the password of the account that is chosen in the second step.

12. Click **OK** and complete the required steps to turn on security.

Results

This set of steps is required to set up the stand-alone custom registry and to enable security in WebSphere Application Server.

Note: The security component of WebSphere Application Server expands a selected list of variables when enabling security. See the information about variable settings for more details.

What to do next

1. Complete the remaining steps, if you are enabling security.
2. Validate the user and password. Save and synchronize in the cell environment.
3. After security is turned on, save, stop, and start all the product servers, including cell, nodes, and all of the application servers, for any changes to take effect. If the server comes up without any problems, the setup is correct.

Stand-alone custom registries:

A *stand-alone custom registry* is a customer-implemented registry that implements the UserRegistry Java interface, as provided by the product. A custom-implemented registry can support virtually any type of an account repository from a relational database, flat file, and so on. The custom user registry provides considerable flexibility in adapting product security to various environments where some form of a registry or repository other than federated repositories, stand-alone Lightweight Directory Access Protocol (LDAP) registry or local operating system registry already exists in the operational environment.

WebSphere Application Server security provides an implementation that uses various local operating system-based registries and various stand-alone Lightweight Directory Access Protocol (LDAP)-based registries. However, situations can exist where your user and group data resides in other repositories or custom user registries, such as a database, and moving this information to either a local operating system registry or a stand-alone LDAP registry implementation might not be feasible. For these situations, WebSphere Application Server security provides a service provider interface (SPI) that you can implement to interact with your current registry. The custom registry feature supports any user registry that is not implemented by WebSphere Application Server.

The SPI is the UserRegistry interface. The *UserRegistry interface* is a collection of methods that are required for authorization purposes. These methods authenticate individual users using either a password or certificates and collect information about the user, which are called privilege attributes. This interface also includes methods that obtain user and group information so that they can be given access to resources. When implementing the methods in the interface, you must decide how to map the information that is manipulated by the UserRegistry interface to the information in your registry.

This interface has a set of methods to implement for the product security to interact with your registries for all security-related tasks. The local operating system and LDAP registry implementations that are provided also implement this interface. Custom user registries are sometimes called the *pluggable user registries* or *custom registries* for short. Your custom user registry implementation is expected to be thread-safe.

Building a custom registry is a software implementation effort. The implementation does not depend on other WebSphere Application Server resources, for example, data sources, for its operation.

Make sure that your implementation of the custom registry does not depend on any WebSphere Application Server components such as data sources, enterprise beans, and so on. Do not have this dependency because security is initialized and enabled prior to most of the other WebSphere Application Server components during startup. If your previous implementation used these components, make a change that eliminates the dependency.

The methods in the UserRegistry interface operate on the following information for users:

User security name
The user name is similar to the user profile in the operating system registry.

This name is used to log in when prompted by a secured application. By default, the Enterprise JavaBeans (EJB) getCallerPrincipal method and the getRemoteUser and getUserPrincipal servlet methods return this name. The user security name is also referred to as *userSecurityName*, *userName*, or *user name*.

WAS_UseDisplayName
This is a custom property of User Registries. This property defines the returning value of the getCallerPrincipal(), getUserPrincipal(), and RemoteUser() methods. The following shows acceptable values for WAS_UseDisplayName:
- `false` This is default. Security Name is returned.
- `true` The display name is returned. This setting requires that the custom property **com.ibm.websphere.security.useLoggedSecurityName** be set to true.

Unique user ID
This ID represents a unique identifier for the user, which is required by the UserRegistry interface. The unique ID is similar to the system ID (SID) in Windows systems, the Unique ID (UID) in Linux and UNIX systems, and the distinguished name (DN) in Lightweight Directory Authentication Protocol (LDAP). This ID is also referred to as *uniqueUserId*. The unique ID is used to make the authorization decisions for protected resources.

Display user name
The display name is the text description for the user profile.

Group security name
This name, which represents the security group, is also referred to as *groupSecurityName*, *groupName*, and *group name*.

Unique group ID
The unique ID is the identifier for a group. This name is also referred to as *uniqueGroupId* ID.

Display group name
The display name is an optional string that describes a group.

The topic on UserRegistry interface describes each of the methods in the interface that need implementing. An explanation of each of the methods and their usage in the sample and any changes from the Version 4 interface are provided. The Related references section provides links to all other custom user registries documentation, including a file-based registry sample. The Sample provided is very simple and is intended to familiarize you with this feature. Do not use this sample in an actual production environment.

Stand-alone custom registry settings:

Use this page to configure the stand-alone custom registry.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Stand-alone custom registry**, and click **Configure**.

After the properties are set in this panel, click **Apply**. Under Additional Properties, click **Custom properties** to include additional properties that the custom user registry requires.

Note: Custom properties might include information such as specifying lists of users or groups.

When security is enabled and any of these custom user registry settings change, go to the Global security panel and click **Apply** to validate the changes.

WebSphere Application Server Version 7.0 distinguishes between the user identities for administrators who manage the environment and server identities for authenticating server to server communications. In most cases, server identities are automatically generated and are not stored in a repository.

Primary administrative user name:

Specifies the name of a user with administrative privileges that is defined in your custom user registry.

The user name is used to log onto the administrative console when administrative security is enabled. Version 6.1 requires an administrative user that is distinct from the server user identity so that administrative actions can be audited.

Attention: In WebSphere Application Server, Version 6.0.x, a single user identity is required for both administrative access and internal process communication. When migrating to Version 6.1, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Automatically generated server identity:

Enables the application server to generate the server identity, which is recommended for environments that contain only Version 6.1 or later nodes. Automatically generated server identities are not stored in a user repository.

Default: Enabled

Server identity that is stored in the repository:

Specifies a user identity in the repository that is used for internal process communication. Cells that contain Version 6.1 or later nodes require a server user identity that is defined in the active user repository.

Default: Enabled

Custom registry class name:

Specifies a dot-separated class name that implements the com.ibm.websphere.security.UserRegistry interface.

Put the custom registry class name in the class path. A suggested location is the following directory.

- *profile_root*/classes

Data type: String
Default: com.ibm.websphere.security.FileRegistrySample

Ignore case for authorization:

Indicates that a case-insensitive authorization check is performed when you use the default authorization.

Default: Disabled
Range: Enabled or Disabled

Stand-alone custom registry wizard settings:

A wizard page exists in the administrative console to aid in viewing the basic settings necessary to connect the application server to an existing stand-alone custom registry. After you have viewed the basic settings, you can also modify the existing stand-alone customer registry configuration using the administrative console.

To view this security wizard page, complete the following steps:

1. Click **Security > Global security > Security configuration wizard**.
2. Select your protection settings and click **Next**.
3. Select the **Stand-alone custom registry** option and click **Next**.

You can modify your stand-alone custom registry configuration by completing the following steps:

1. Click **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Stand-alone custom registry**, and click **Configure**.
3. Enter additional properties to initialize your implementation
   - Click **Custom properties > New**.
   - Enter the property name and value. For the sample, enter the following two properties. It is assumed that the `users.props` file and the `groups.props` file are in the *customer_sample* directory under the product installation directory. You can place these properties in any directory that you choose and reference their locations through Custom properties. However, make sure that the directory has the appropriate access permissions.

Table 83. Custom properties.

This table lists additional custom properties when changing stand-alone custom registry wizard settings.

Property name	Property value
usersFile	${*USER_INSTALL_ROOT*}/*customer_sample* /users.props
groupsFile	${*USER_INSTALL_ROOT*}/*customer_sample* /groups.props

     Samples of these two properties are available in reference topics for the `users.props` file and the `groups.props` file. See the related links below for more information.

     The Description, Required, and Validation Expression fields are not used and can remain blank.

     WebSphere Application Server Version 4 based custom user registry is migrated to the custom user registry based on the com.ibm.websphere.security.UserRegistry interface.
   - Click **Apply**.

Primary administrative user name:

Specifies the name of a user with administrative privileges that is defined in your custom user registry.

The user name is used to log onto the administrative console when administrative security is enabled. Version 6.1 requires an administrative user that is distinct from the server user identity so that administrative actions can be audited.

Attention: In WebSphere Application Server, Version 6.0.x, a single user identity is required for both administrative access and internal process communication. When migrating to Version 6.1, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Custom registry class name:

Specifies a dot-separated class name that implements the com.ibm.websphere.security.UserRegistry interface.

Put the custom registry class name in the class path. A suggested location is the following directory.

- *profile_root*/classes

Data type: String
Default: com.ibm.websphere.security.FileRegistrySample

Ignore case for authorization:

Indicates that a case-insensitive authorization check is performed when you use the default authorization.

Default: Disabled
Range: Enabled or Disabled

FileRegistrySample.java file:

This provides an example of the FileRegistrySample.java file.

The user and group information required by this sample is contained in the "users.props file" on page 1310 and "groups.props file" on page 1310 files.

Attention: The samples that are provided are intended to familiarize you with this feature. Do not use these samples in an actual production environment.

The contents of the FileRegistrySample.java file:

```
//
// 5639-D57, 5630-A36, 5630-A37, 5724-D18
// (C) COPYRIGHT International Business Machines Corp. 1997, 2005
// All Rights Reserved * Licensed Materials - Property of IBM
////--------------------------------------------------------------------
// This program may be used, run, copied, modified and distributed
// without royalty for the purpose of developing, using, marketing, or
// distributing.
//--------------------------------------------------------------------
//

// This sample is for the custom user registry feature in WebSphere Application Server.

import java.util.*;
import java.io.*;
import java.security.cert.X509Certificate;
import com.ibm.websphere.security.*;
/**
 *  The main purpose of this sample is to demonstrate the use of the
 *  custom user registry feature available in WebSphere Application Server. This
 *  sample is a file-based registry sample where the users and the groups
 *  information is listed in files (users.props and groups.props). As such
 *  simplicity and not the performance was a major factor. This
 *  sample should be used only to get familiarized with this feature. An
 *  actual implementation of a realistic registry should consider various
 *  factors like performance, scalability, thread safety, and so on.
 **/
public class FileRegistrySample implements UserRegistry {

   private static String USERFILENAME = null;
   private static String GROUPFILENAME = null;

   /** Default Constructor **/
   public FileRegistrySample() throws java.rmi.RemoteException {
```

```
 }

/**
 * Initializes the registry. This method is called when creating the
 * registry.
 *
 * @param     props - The registry-specific properties with which to
 *                    initialize the custom registry
 * @exception CustomRegistryException
 *                    if there is any registry-specific problem
 **/
public void initialize(java.util.Properties props)
       throws CustomRegistryException {
    try {
       /* try getting the USERFILENAME and the GROUPFILENAME from
        * properties that are passed in (For example, from the
        * administrative console). Set these values in the administrative
        * console. Go to the special custom settings in the custom
        * user registry section of the Authentication panel.
        * For example:
        * usersFile   c:/temp/users.props
        * groupsFile  c:/temp/groups.props
        */
        if (props != null) {
           USERFILENAME = props.getProperty("usersFile");
           GROUPFILENAME = props.getProperty("groupsFile");
        }

    } catch(Exception ex) {
       throw new CustomRegistryException(ex.getMessage(),ex);
    }

    if (USERFILENAME == null || GROUPFILENAME == null) {
       throw new CustomRegistryException("users/groups information missing");
    }

 }  /**
 * Checks the password of the user. This method is called to authenticate
 * a user when the user's name and password are given.
 *
 * @param  userSecurityName the name of user
 * @param  password the password of the user
 * @return a valid userSecurityName. Normally this is
 *         the name of same user whose password was checked
 *         but if the implementation wants to return any other
 *         valid userSecurityName in the registry it can do so
 * @exception CheckPasswordFailedException if userSecurityName/
 *            password combination does not exist
 *            in the registry
 * @exception CustomRegistryException if there is any registry-
 *            specific problem
 **/
 public String checkPassword(String userSecurityName, String passwd)
    throws PasswordCheckFailedException,
           CustomRegistryException {
    String s,userName = null;
    BufferedReader in = null;

    try {
       in = fileOpen(USERFILENAME);
       while ((s=in.readLine())!=null)
       {
          if (!(s.startsWith("#") || s.trim().length() <=0 )) {
```

```
            int index = s.indexOf(":");
            int index1 = s.indexOf(":",index+1);
            // check if the userSecurityName:passwd combination exists
            if ((s.substring(0,index)).equals(userSecurityName) &&
                    s.substring(index+1,index1).equals(passwd)) {
               // Authentication successful, return the userID.
               userName = userSecurityName;
               break;
            }
         }
      }
   } catch(Exception ex) {
      throw new CustomRegistryException(ex.getMessage(),ex);
   } finally {
      fileClose(in);
   }


   if (userName == null) {
      throw new PasswordCheckFailedException("Password check failed for user: "
      + userSecurityName);
   }

   return userName;
}  /**
* Maps an X.509 format certificate to a valid user in the registry.
* This is used to map the name in the certificate supplied by a browser
* to a valid userSecurityName in the registry
*
* @param     cert the X509 certificate chain
* @return    The mapped name of the user userSecurityName
* @exception CertificateMapNotSupportedException if the
*            particular certificate is not supported.
* @exception CertificateMapFailedException if the mapping of
*            the certificate fails.
* @exception CustomRegistryException if there is any registry
*            -specific problem
**/
public String mapCertificate(X509Certificate[] cert)
   throws CertificateMapNotSupportedException,
          CertificateMapFailedException,
          CustomRegistryException {
   String name=null;
   X509Certificate cert1 = cert[0];
   try {
      // map the SubjectDN in the certificate to a userID.
      name = cert1.getSubjectDN().getName();
   } catch(Exception ex) {
      throw new CertificateMapNotSupportedException(ex.getMessage(),ex);
   }

   if(!isValidUser(name)) {
      throw new CertificateMapFailedException("user: " + name
      + " is not valid");
   }
   return name;
}  /**
* Returns the realm of the registry.
*
* @return the realm. The realm is a registry-specific string
* indicating the realm or domain for which this registry
* applies. For example, for OS/400 or AIX this would be
* the host name of the system whose user registry this
```

```
* object represents. If null is returned by this method,
* realm defaults to the value of "customRealm". It is
* recommended that you use your own value for realm.
*
* @exception CustomRegistryException if there is any registry-
* specific problem
**/
public String getRealm()
   throws CustomRegistryException {
   String name = "customRealm";
   return name;
}  /**
* Gets a list of users that match a pattern in the registry.
* The maximum number of users returned is defined by the limit
* argument.
* This method is called by the administrative console and scripting
* (command line) to make the users in the registry available for
* adding them (users) to roles.
*
* @param      pattern the pattern to match. (For example, a* will
*             match  all userSecurityNames starting with a)
* @param      limit the maximum number of users that should be
*             returned. This is very useful in situations where
*             there are thousands of users in the registry and
*             getting all of them at once is not practical. The
*             default is 100. A value of 0 implies get all the
*             users and hence must be used with care.
* @return     a Result object that contains the list of users
*             requested and a flag to indicate if  more users
*             exist.
* @exception  CustomRegistryException if there is any registry-
*             specific problem
**/
public Result getUsers(String pattern, int limit)
   throws CustomRegistryException {
   String s;
   BufferedReader in = null;
   List allUsers = new ArrayList();
   Result result = new Result();
   int count = 0;
   int newLimit = limit+1;
   try {
      in = fileOpen(USERFILENAME);
      while ((s=in.readLine())!=null)
      {
         if (!(s.startsWith("#") || s.trim().length() <=0 )) {
            int index = s.indexOf(":");
            String user = s.substring(0,index);
            if (match(user,pattern)) {
               allUsers.add(user);
               if (limit !=0 && ++count == newLimit) {
                  allUsers.remove(user);
                  result.setHasMore();
                  break;
               }
            }
         }
      }
   } catch (Exception ex) {
      throw new CustomRegistryException(ex.getMessage(),ex);
   } finally {
      fileClose(in);
   }
```

```
   result.setList(allUsers);
   return result;
}  /**
* Returns the display name for the user specified by
*  userSecurityName.
*
* This method may be called only when the user information
* is displayed (information purposes only, for example, in
* the administrative console) and hence not used in the actual
* authentication or authorization purposes. If there are no
* display names in the registry return null or empty string.
*
* In WebSphere Application Server 4.x custom registry, if you
* had a display name for the user and if it was different from the
* security name, the display name was returned for the EJB
* methods getCallerPrincipal() and the servlet methods
* getUserPrincipal() and  getRemoteUser().
* In WebSphere Application Server Version 5.x and later, for the
* same methods, the security name will be returned by default.
* This is the recommended way as the display name is not unique
* and might create security holes. However, for backward
* compatibility if you need the display name to be returned
* set the property WAS_UseDisplayName to true.
*
*See the Information Center documentation for more information.
*
* @param     userSecurityName the name of the user.
* @return    the display name for the user. The display
*            name is a registry-specific string that
*            represents a descriptive, not necessarily
*            unique, name for a user. If a display name
*            does not exist return null or empty string.
* @exception EntryNotFoundException if userSecurityName
*            does not exist.
* @exception CustomRegistryException if there is any registry-
 *           specific problem
**/
public String getUserDisplayName(String userSecurityName)
   throws CustomRegistryException,
          EntryNotFoundException {

   String s,displayName = null;
   BufferedReader in = null;

   if(!isValidUser(userSecurityName)) {
      EntryNotFoundException nsee = new EntryNotFoundException("user: "
      + userSecurityName + " is not valid");
      throw nsee;
   }

   try {
      in = fileOpen(USERFILENAME);
      while ((s=in.readLine())!=null)
      {
         if (!(s.startsWith("#") || s.trim().length() <=0 )) {
            int index = s.indexOf(":");
            int index1 = s.lastIndexOf(":");
            if ((s.substring(0,index)).equals(userSecurityName)) {
               displayName = s.substring(index1+1);
               break;
            }
         }
```

```
        }
     } catch(Exception ex) {
        throw new CustomRegistryException(ex.getMessage(), ex);
     } finally {
        fileClose(in);
     }

     return displayName;
  }

 /**
  * Returns the unique ID for a userSecurityName. This method is called
  * when creating a credential for a user.
  *
  * @param    userSecurityName - The name of the user.
  * @return   The unique ID of the user. The unique ID for a user
  *           is the stringified form of some unique, registry-specific,
  *           data that serves to represent the user. For example, for
  *           the UNIX user registry, the unique ID for a user can be
  *           the UID.
  * @exception EntryNotFoundException if userSecurityName does not
  *            exist.
  * @exception CustomRegistryException if there is any registry-
  *            specific problem
  **/
  public String getUniqueUserId(String userSecurityName)
     throws CustomRegistryException,
            EntryNotFoundException {

     String s,uniqueUsrId = null;
     BufferedReader in = null;
     try {
        in = fileOpen(USERFILENAME);
        while ((s=in.readLine())!=null)
        {
           if (!(s.startsWith("#") || s.trim().length() <=0 )) {
              int index = s.indexOf(":");
              int index1 = s.indexOf(":", index+1);
              if ((s.substring(0,index)).equals(userSecurityName)) {
                 int index2 = s.indexOf(":", index1+1);
                 uniqueUsrId = s.substring(index1+1,index2);
                 break;
              }
           }
        }
     } catch(Exception ex) {
        throw new CustomRegistryException(ex.getMessage(),ex);
     } finally {
        fileClose(in);
     }

     if (uniqueUsrId == null) {
        EntryNotFoundException nsee =
        new EntryNotFoundException("Cannot obtain uniqueId for user: "
        + userSecurityName);
        throw nsee;
     }

     return uniqueUsrId;
  }  /**
  * Returns the name for a user given its unique ID.
  *
  * @param      uniqueUserId  - The unique ID of the user.
```

```
* @return     The userSecurityName of the user.
* @exception  EntryNotFoundException if the unique user ID does not exist.
* @exception  CustomRegistryException if there is any registry-specific
*             problem
**/
public String getUserSecurityName(String uniqueUserId)
   throws CustomRegistryException,
          EntryNotFoundException {
   String s,usrSecName = null;
   BufferedReader in = null;
   try {
      in = fileOpen(USERFILENAME);
      while ((s=in.readLine())!=null)
      {
         if (!(s.startsWith("#") || s.trim().length() <=0 )) {
            int index = s.indexOf(":");
            int index1 = s.indexOf(":", index+1);
            int index2 = s.indexOf(":", index1+1);
            if ((s.substring(index1+1,index2)).equals(uniqueUserId)) {
               usrSecName = s.substring(0,index);
               break;
            }
         }
      }
   } catch (Exception ex) {
      throw new CustomRegistryException(ex.getMessage(), ex);
   } finally {
      fileClose(in);
   }

   if (usrSecName == null) {
      EntryNotFoundException ex =
         new EntryNotFoundException("Cannot obtain the
         user securityName for " + uniqueUserId);
      throw ex;
   }

   return usrSecName;

}  /**
* Determines if the userSecurityName exists in the registry
*
* @param     userSecurityName - The name of the user
* @return    True if the user is valid; otherwise false
* @exception CustomRegistryException if there is any registry-
*            specific problem
* @exception RemoteException as this extends java.rmi.Remote
*            interface
**/
public boolean isValidUser(String userSecurityName)
   throws CustomRegistryException {
   String s;
   boolean isValid = false;
   BufferedReader in = null;
   try {
      in = fileOpen(USERFILENAME);
      while ((s=in.readLine())!=null)
      {
         if (!(s.startsWith("#") || s.trim().length() <=0 )) {
            int index = s.indexOf(":");
            if ((s.substring(0,index)).equals(userSecurityName)) {
               isValid=true;
               break;
```

```
            }
         }
      }
   } catch (Exception ex) {
      throw new CustomRegistryException(ex.getMessage(), ex);
   } finally {
      fileClose(in);
   }

   return isValid;
}
/**
 * Gets a list of groups that match a pattern in the registry
 * The maximum number of groups returned is defined by the
 * limit argument. This method is called by administrative console
 * and scripting (command line) to make available the groups in
 * the registry for adding them (groups) to roles.
 *
 * @param       pattern the pattern to match. (For example, a* matches
 *              all groupSecurityNames starting with a)
 * @param       Limits the maximum number of groups to return
 *              This is very useful in situations where there
 *              are thousands of groups in the registry and getting all
 *              of them at once is not practical. The default is 100.
 *              A value of 0 implies get all the groups and hence must
 *              be used with care.
 * @return      A Result object that contains the list of groups
 *              requested and a flag to indicate if more groups exist.
 * @exception CustomRegistryException if there is any registry-specific
 *              problem
 **/
public Result getGroups(String pattern, int limit)
   throws CustomRegistryException {
   String s;
   BufferedReader in = null;
   List allGroups = new ArrayList();      Result result = new Result();
   int count = 0;
   int newLimit = limit+1;
   try {
      in = fileOpen(GROUPFILENAME);
      while ((s=in.readLine())!=null)
      {
         if (!(s.startsWith("#") || s.trim().length() <=0 )) {
            int index = s.indexOf(":");
            String group = s.substring(0,index);
            if (match(group,pattern)) {
               allGroups.add(group);
               if (limit !=0 && ++count == newLimit) {
                  allGroups.remove(group);
                  result.setHasMore();
                  break;
               }
            }
         }
      }
   } catch (Exception ex) {
      throw new CustomRegistryException(ex.getMessage(),ex);
   } finally {
      fileClose(in);
   }

   result.setList(allGroups);
   return result;
```

```
 }

/**
 * Returns the display name for the group specified by groupSecurityName.
 * For this version of WebSphere Application Server, the only usage of
 * this method is by the clients (administrative console and scripting)
 * to present a descriptive name of the user if it exists.
 *
 * @param groupSecurityName the name of the group.
 * @return  the display name for the group. The display name
 *          is a registry-specific string that represents a
 *          descriptive, not necessarily unique, name for a group.
 *          If a display name does not exist return null or empty
 *          string.
 * @exception EntryNotFoundException if groupSecurityName does
 *          not exist.
 * @exception CustomRegistryException if there is any registry-
 *          specific problem
 **/
 public String getGroupDisplayName(String groupSecurityName)
    throws CustomRegistryException,
           EntryNotFoundException {
    String s,displayName = null;
    BufferedReader in = null;

    if(!isValidGroup(groupSecurityName)) {
       EntryNotFoundException nsee = new EntryNotFoundException("group: "
       + groupSecurityName + " is not valid");
       throw nsee;
    }

    try {
       in = fileOpen(GROUPFILENAME);
       while ((s=in.readLine())!=null)
       {
          if (!(s.startsWith("#") || s.trim().length() <=0 )) {
             int index = s.indexOf(":");
             int index1 = s.lastIndexOf(":");
             if ((s.substring(0,index)).equals(groupSecurityName)) {
                displayName = s.substring(index1+1);
                break;
             }
          }
       }
    } catch(Exception ex) {
       throw new CustomRegistryException(ex.getMessage(),ex);
    } finally {
       fileClose(in);
    }

    return displayName;
 }

/**
 * Returns the Unique ID for a group.

 * @param     groupSecurityName the name of the group.
 * @return    The unique ID of the group. The unique ID for
 *            a group is the stringified form of some unique,
 *            registry-specific, data that serves to represent
 *            the group. For example, for the UNIX user registry,
 *            the unique ID might be the GID.
 * @exception EntryNotFoundException if groupSecurityName does
```

```
 *            not exist.
 * @exception CustomRegistryException if there is any registry-
 *            specific problem
 * @exception RemoteException as this extends java.rmi.Remote
 **/
 public String getUniqueGroupId(String groupSecurityName)
    throws CustomRegistryException,
           EntryNotFoundException {
    String s,uniqueGrpId = null;
    BufferedReader in = null;
    try {
       in = fileOpen(GROUPFILENAME);
       while ((s=in.readLine())!=null)
       {
          if (!(s.startsWith("#") || s.trim().length() <=0 )) {
             int index = s.indexOf(":");
             int index1 = s.indexOf(":", index+1);
             if ((s.substring(0,index)).equals(groupSecurityName)) {
                uniqueGrpId = s.substring(index+1,index1);
                break;
             }
          }
       }
    } catch(Exception ex) {
       throw new CustomRegistryException(ex.getMessage(),ex);
    } finally {
       fileClose(in);
    }

    if (uniqueGrpId == null) {
       EntryNotFoundException nsee =
       new EntryNotFoundException("Cannot obtain the uniqueId for group: "
       + groupSecurityName);
       throw nsee;
    }

    return uniqueGrpId;
 }

/**
 * Returns the Unique IDs for all the groups that contain the unique ID
 * of a user. Called during creation of a user's credential.
 *
 * @param     uniqueUserId the unique ID of the user.
 * @return    A list of all the group unique IDs that the unique user
 *            ID belongs to. The unique ID for an entry is the
 *            stringified form of some unique, registry-specific, data
 *            that serves to represent the entry.  For example, for the
 *            UNIX user registry, the unique ID for a group might be
 *            the GID and the Unique ID for the user might be the UID.
 * @exception EntryNotFoundException if uniqueUserId does not exist.
 * @exception CustomRegistryException if there is any registry-specific
 *            problem
 **/
 public List getUniqueGroupIds(String uniqueUserId)
    throws CustomRegistryException,
           EntryNotFoundException {
    String s,uniqueGrpId = null;
    BufferedReader in = null;
    List uniqueGrpIds=new ArrayList();
    try {
       in = fileOpen(USERFILENAME);
       while ((s=in.readLine())!=null)
```

```
      {
         if (!(s.startsWith("#") || s.trim().length() <=0 )) {
            int index = s.indexOf(":");
            int index1 = s.indexOf(":", index+1);
            int index2 = s.indexOf(":", index1+1);
            if ((s.substring(index1+1,index2)).equals(uniqueUserId)) {
               int lastIndex = s.lastIndexOf(":");
               String subs = s.substring(index2+1,lastIndex);
               StringTokenizer st1 = new StringTokenizer(subs, ",");
               while (st1.hasMoreTokens())
                  uniqueGrpIds.add(st1.nextToken());
               break;
            }
         }
      }
   } catch(Exception ex) {
      throw new CustomRegistryException(ex.getMessage(),ex);
   } finally {
      fileClose(in);
   }

   return uniqueGrpIds;
}

/**
 * Returns the name for a group given its unique ID.
 *
 * @param     uniqueGroupId the unique ID of the group.
 * @return    The name of the group.
 * @exception EntryNotFoundException if the uniqueGroupId does
 *            not exist.
 * @exception CustomRegistryException if there is any registry-
 *            specific problem
 **/
public String getGroupSecurityName(String uniqueGroupId)
   throws CustomRegistryException,
          EntryNotFoundException {
   String s,grpSecName = null;
   BufferedReader in = null;
   try {
      in = fileOpen(GROUPFILENAME);
      while ((s=in.readLine())!=null)
      {
         if (!(s.startsWith("#") || s.trim().length() <=0 )) {
            int index = s.indexOf(":");
            int index1 = s.indexOf(":", index+1);
            if ((s.substring(index+1,index1)).equals(uniqueGroupId)) {
               grpSecName = s.substring(0,index);
               break;
            }
         }
      }
   } catch (Exception ex) {
      throw new CustomRegistryException(ex.getMessage(),ex);
   } finally {
      fileClose(in);
   }

   if (grpSecName == null) {
      EntryNotFoundException ex =
         new EntryNotFoundException("Cannot obtain the group
        security name for: " + uniqueGroupId);
      throw ex;
```

```
    }

    return grpSecName;

 }

/**
 * Determines if the groupSecurityName exists in the registry
 *
 * @param     groupSecurityName the name of the group
 * @return    True if the groups exists; otherwise false
 * @exception CustomRegistryException if there is any registry-
 *            specific problem
 **/
 public boolean isValidGroup(String groupSecurityName)
    throws CustomRegistryException {
    String s;
    boolean isValid = false;
    BufferedReader in = null;
    try {
       in = fileOpen(GROUPFILENAME);
       while ((s=in.readLine())!=null)
       {
          if (!(s.startsWith("#") || s.trim().length() <=0 )) {
             int index = s.indexOf(":");
             if ((s.substring(0,index)).equals(groupSecurityName)) {
                isValid=true;
                break;
             }
          }
       }
    } catch (Exception ex) {
       throw new CustomRegistryException(ex.getMessage(),ex);
    } finally {
       fileClose(in);
    }

    return isValid;
 }

/**
 * Returns the securityNames of all the groups that contain the user
 *
 * This method is called by the administrative console and scripting
 * (command line) to verify that the user entered for RunAsRole mapping
 * belongs to that role in the roles to user mapping. Initially, the
 * check is done to see  if the role contains the user. If the role does
 * not contain the user explicitly, this method is called to get the groups
 * that this user belongs to so that a check can be made on the groups that
 * the role contains.
 *
 * @param     userSecurityName the name of the user
 * @return    A list of all the group securityNames that the user
 *            belongs to.
 * @exception EntryNotFoundException if user does not exist.
 * @exception CustomRegistryException if there is any registry-
 *            specific problem
 * @exception RemoteException as this extends the java.rmi.Remote
 *            interface
 **/
 public List getGroupsForUser(String userName)
    throws CustomRegistryException,
           EntryNotFoundException {
```

```
    String s;
    List grpsForUser = new ArrayList();
    BufferedReader in = null;
    try {
       in = fileOpen(GROUPFILENAME);
       while ((s=in.readLine())!=null)
       {
          if (!(s.startsWith("#") || s.trim().length() <=0 )) {
             StringTokenizer st = new StringTokenizer(s, ":");
             for (int i=0; i<2; i++)
                st.nextToken();
             String subs = st.nextToken();
             StringTokenizer st1 = new StringTokenizer(subs, ",");
             while (st1.hasMoreTokens()) {
                if((st1.nextToken()).equals(userName)) {
                   int index = s.indexOf(":");
                   grpsForUser.add(s.substring(0,index));
                }
             }
          }
       }
    } catch (Exception ex) {
       if (!isValidUser(userName)) {
          throw new EntryNotFoundException("user: " + userName
          + " is not valid");
       }
       throw new CustomRegistryException(ex.getMessage(),ex);
    } finally {
       fileClose(in);
    }

    return grpsForUser;
 }

/**
 * Gets a list of users in a group.
 *
 * The maximum number of users returned is defined by the
 * limit argument.
 *
 * This method is being used by the WebSphere Application Server
 * Enterprise process choreographer (Enterprise) when
 * staff assignments are modeled using groups.
 *
 * In rare situations, if you are working with a registry where
 * getting all the users from any of your groups is not practical
 * (for example if there are a large number of users) you can create
 * the NotImplementedException for that particular group. Make sure
 * that if the process choreographer is installed (or if installed later)
 * the staff assignments are not modeled using these particular groups.
 * If there is no concern about returning the users from groups
 * in the registry it is recommended that this method be implemented
 * without creating the NotImplemented exception.
 * @param         groupSecurityName the name of the group
 * @param         Limits the maximum number of users that should be
 *                returned. This is very useful in situations where there
 *                are lots of users in the registry and getting all of
 *                them at once is not practical. A value of 0 implies
 *                get all the users and hence must be used with care.
 * @return        A Result object that contains the list of users
 *                requested and a flag to indicate if more users exist.
 * @deprecated    This method will be deprecated in future.
 * @exception     NotImplementedException create this exception in rare
```

```
 *                situations if it is not practical to get this information
 *                for any of the group or groups from the registry.
 * @exception     EntryNotFoundException if the group does not exist in
 *                the registry
 * @exception     CustomRegistryException if there is any registry-specific
 *                problem
 **/
 public Result getUsersForGroup(String groupSecurityName, int limit)
    throws NotImplementedException,
           EntryNotFoundException,
           CustomRegistryException {
    String s, user;
    BufferedReader in = null;
    List usrsForGroup = new ArrayList();
    int count = 0;
    int newLimit = limit+1;
    Result result = new Result();

    try {
       in = fileOpen(GROUPFILENAME);
       while ((s=in.readLine())!=null)
       {
          if (!(s.startsWith("#") || s.trim().length() <=0 )) {
             int index = s.indexOf(":");
             if ((s.substring(0,index)).equals(groupSecurityName))
             {
                StringTokenizer st = new StringTokenizer(s, ":");
                for (int i=0; i<2; i++)
                   st.nextToken();
                String subs = st.nextToken();
                StringTokenizer st1 = new StringTokenizer(subs, ",");
                while (st1.hasMoreTokens()) {
                   user = st1.nextToken();
                   usrsForGroup.add(user);
                   if (limit !=0 && ++count == newLimit) {
                      usrsForGroup.remove(user);
                      result.setHasMore();
                      break;
                   }
                }
             }
          }
       }
    } catch (Exception ex) {
       if (!isValidGroup(groupSecurityName)) {
          throw new EntryNotFoundException("group: "
          + groupSecurityName
          + " is not valid");
       }
       throw new CustomRegistryException(ex.getMessage(),ex);
    } finally {
       fileClose(in);
    }

    result.setList(usrsForGroup);
    return result;
 }

/**
 * This method is implemented internally by the WebSphere Application Server
 * code in this release. This method is not called for the custom
 * registry implementations for this release. Return null in the
 * implementation.
```

```
   *
   **/
   public com.ibm.websphere.security.cred.WSCredential
         createCredential(String userSecurityName)
         throws CustomRegistryException,
                NotImplementedException,
                EntryNotFoundException {

      // This method is not called.
      return null;
   }

   // private methods
   private BufferedReader fileOpen(String fileName)
      throws FileNotFoundException {
      try {
         return new BufferedReader(new FileReader(fileName));
      } catch(FileNotFoundException e) {
         throw e;
      }
   }

   private void fileClose(BufferedReader in) {
      try {
         if (in != null) in.close();
      } catch(Exception e) {
         System.out.println("Error closing file" + e);
      }
   }

   private boolean match(String name, String pattern) {
      RegExpSample regexp = new RegExpSample(pattern);
      boolean matches = false;
      if(regexp.match(name))
          matches = true;
      return matches;
   }
}


//--------------------------------------------------------------------------
// The program provides the Regular Expression implementation
// used in the sample for the custom user registry (FileRegistrySample).
// The pattern matching in the sample uses this program to search for the
// pattern (for users and groups).
//--------------------------------------------------------------------------

class RegExpSample
{

    private boolean match(String s, int i, int j, int k)
    {
        for(; k < expr.length; k++)
label0:
            {
                Object obj = expr[k];
                if(obj == STAR)
                {
                    if(++k >= expr.length)
                        return true;
                    if(expr[k] instanceof String)
                    {
                        String s1 = (String)expr[k++];
```

```
                    int l = s1.length();
                    for(; (i = s.indexOf(s1, i)) >= 0; i++)
                        if(match(s, i + l, j, k))
                            return true;

                    return false;
                }
                for(; i < j; i++)
                    if(match(s, i, j, k))
                        return true;

                return false;
            }
            if(obj == ANY)
            {
                if(++i > j)
                    return false;
                break label0;
            }
            if(obj instanceof char[][])
            {
                if(i >= j)
                    return false;
                char c = s.charAt(i++);
                char ac[][] = (char[][])obj;
                if(ac[0] == NOT)
                {
                    for(int j1 = 1; j1 < ac.length; j1++)
                        if(ac[j1][0] <= c && c <= ac[j1][1])
                            return false;

                    break label0;
                }
                for(int k1 = 0; k1 < ac.length; k1++)
                    if(ac[k1][0] <= c && c <= ac[k1][1])
                        break label0;

                return false;
            }
            if(obj instanceof String)
            {
                String s2 = (String)obj;
                int i1 = s2.length();
                if(!s.regionMatches(i, s2, 0, i1))
                    return false;
                i += i1;
            }
        }

    return i == j;
}

public boolean match(String s)
{
    return match(s, 0, s.length(), 0);
}

public boolean match(String s, int i, int j)
{
    return match(s, i, j, 0);
}

public RegExpSample(String s)
```

```
{
    Vector vector = new Vector();
    int i = s.length();
    StringBuffer stringbuffer = null;
    Object obj = null;
    for(int j = 0; j < i; j++)
    {
        char c = s.charAt(j);
        switch(c)
        {
        case 63: /* '?' */
            obj = ANY;
            break;

        case 42: /* '*' */
            obj = STAR;
            break;

        case 91: /* '[' */
            int k = ++j;
            Vector vector1 = new Vector();
            for(; j < i; j++)
            {
                c = s.charAt(j);
                if(j == k && c == '^')
                {
                    vector1.addElement(NOT);
                    continue;
                }
                if(c == '\\')
                {
                    if(j + 1 < i)
                        c = s.charAt(++j);
                }
                else
                if(c == ']')
                    break;
                char c1 = c;
                if(j + 2 < i && s.charAt(j + 1) == '-')
                    c1 = s.charAt(j += 2);
                char ac1[] = {
                    c, c1
                };
                vector1.addElement(ac1);
            }

            char ac[][] = new char[vector1.size()][];
            vector1.copyInto(ac);
            obj = ac;
            break;

        case 92: /* '\\' */
            if(j + 1 < i)
                c = s.charAt(++j);
            break;

        }
        if(obj != null)
        {
            if(stringbuffer != null)
            {
                vector.addElement(stringbuffer.toString());
                stringbuffer = null;
```

```
                }
                vector.addElement(obj);
                obj = null;
            }
            else
            {
                if(stringbuffer == null)
                    stringbuffer = new StringBuffer();
                stringbuffer.append(c);
            }
        }

        if(stringbuffer != null)
            vector.addElement(stringbuffer.toString());
        expr = new Object[vector.size()];
        vector.copyInto(expr);
    }

    static final char NOT[] = new char[2];
    static final Integer ANY = new Integer(0);
    static final Integer STAR = new Integer(1);
    Object expr[];

}
```

users.props file:

This example presents the format for the `users.props` file.

Attention: The sample that is provided is intended to familiarize you with this feature. Do not use this sample in an actual production environment.

```
# 5639-D57, 5630-A36, 5630-A37, 5724-D18
# (C) COPYRIGHT International Business Machines Corp. 1997, 2005
# All Rights Reserved * Licensed Materials - Property of IBM
#
# Format:
# name:passwd:uid:gids:display name
# where name   = userId/userName of the user
#       passwd = password of the user
#       uid    = uniqueId of the user
#       gid    = groupIds of the groups that the user belongs to
#       display name = a (optional) display name for the user.
bob:bob1:123:567:bob
dave:dave1:234:678:
jay:jay1:345:678,789:Jay-Jay
ted:ted1:456:678:Teddy G
jeff:jeff1:222:789:Jeff
vikas:vikas1:333:789:vikas
bobby:bobby1:444:789:
```

groups.props file:

The following example illustrates the format for the `groups.props` file.

Attention: The sample provided is intended to familiarize you with this feature. Do not use this sample in an actual production environment.

```
# 5639-D57, 5630-A36, 5630-A37, 5724-D18
# (C) COPYRIGHT International Business Machines Corp. 1997, 2005
# All Rights Reserved * Licensed Materials - Property of IBM
#
# Format:
# name:gid:users:display name
# where name   = groupId of the group
#       gid    = uniqueId of the group
#       users  = list of all the userIds that the group contains
#       display name = a (optional) display name for the group.
admins:567:bob:Administrative group
operators:678:jay,ted,dave:Operators group
users:789:jay,jeff,vikas,bobby:
```

Developing the UserRegistry interface for using custom registries:

Implementing this interface enables WebSphere Application Server security to use custom registries. This capability extends the `java.rmi` file. With a remote registry, you can complete this process remotely.

About this task

Provide implementations of the following methods.

Procedure

- Initialize the UserRegistry method, with initialize(java.util.Properties).

```
public void initialize(java.util.Properties props)
       throws CustomRegistryException,
              RemoteException;
```

  This method is called to initialize the UserRegistry method. All the properties that are defined in the Custom User Registry panel propagate to this method.

  For the `FileRegistrySample.java` sample file, the initialize method retrieves the names of the registry files that contain the user and group information.

  This method is called during server bringup to initialize the registry. This method is also called when validation is performed by the administrative console, when security is on. This method remains the same as in Version 4.x.

- Authenticate users with checkPassword(String,String).

```
public String checkPassword(String userSecurityName, String password)
      throws PasswordCheckFailedException
             CustomRegistryException,
             RemoteException;
```

  The checkPassword method is called to authenticate users when they log in using a name or user ID and a password. This method returns a string which, in most cases, is the user security name. A credential is created for the user for authorization purposes. This user name is also returned for the getCallerPrincipal enterprise bean call and the servlet calls the getUserPrincipal and getRemoteUser methods. See the getUserDisplayName method for more information if you have display names in your registry. In some situations, if you return a user other than the one who is logged in, you must verify that the user is valid in the registry.

  For the `FileRegistrySample.java` sample file, the mapCertificate method gets the distinguished name (DN) from the certificate chain and makes sure it is a valid user in the registry before returning the user. For the sample, the checkPassword method checks the name and password combination in the user registry and, if they match, the method returns the user being authenticated.

  This method is called for various scenarios, for example, by the administrative console to validate the user information after the user registry is initialized. This method is also called when you access protected resources in the product for authenticating the user and before proceeding with the authorization. This method is the same as in Version 4.x.

- Obtain user names from X.509 certificates with mapCertificate(X509Certificate[]).

```
public String mapCertificate(X509Certificate[] cert)
          throws CertificateMapNotSupportedException,
                 CertificateMapFailedException,
                 CustomRegistryException,
                 RemoteException;
```

  The mapCertificate method is called to obtain a user name from an X.509 certificate chain that is supplied by the browser. The complete certificate chain is passed to this method and the implementation can validate the chain if needed and get the user information. A credential is created for this user for authorization purposes. If browser certificates are not supported in your configuration, you can create the CertificateMapNotSupportedException exception. The consequence of not supporting certificates is authentication failure if the challenge type is certificates, even if valid certificates are in the browser.

This method is called when certificates are provided for authentication. For web applications, when the authentication constraints are set to CLIENT-CERT in the `web.xml` file of the application, this method is called to map a certificate to a valid user in the registry. For Java clients, this method is called to map the client certificates in the transport layer, when using transport layer authentication. When the identity assertion token, using the CSIv2 authentication protocol, is set to contain certificates, this method is called to map the certificates to a valid user.

In WebSphere Application Server Version 4.x, the input parameter is the X509Certificate certificate. In WebSphere Application Server Version 5.x and later, this parameter changes to accept an array of X509Certificate certificates such as a certificate chain. In Version 4.x, this parameter is called for web applications only, but in version 5.x and later, you can call this method for both web and Java clients.

- Obtain the security realm name with getRealm.

```
public String getRealm()
       throws CustomRegistryException,
       RemoteException;
```

The getRealm method is called to get the name of the security realm. The name of the realm identifies the security domain for which the registry authenticates users. If this method returns a null value, a customRealm default name is used.

For the `FileRegistrySample.java` sample file, the getRealm method returns the customRealm string. One of the calls to this method occurs when the user registry information is validated. This method is the same method as in Version 4.x.

- Obtain the list of users from the registry with getUsers(String,int).

```
public Result getUsers(String pattern, int limit)
     throws CustomRegistryException,
            RemoteException;
```

The getUsers method returns the list of users from the registry. The names of users depend on the pattern parameter. The number of users are limited by the limit parameter. In a registry that has many users, getting all the users is not practical. So the limit parameter is introduced to limit the number of users retrieved from the registry. A limit of zero (0) indicates to return all the users that match the pattern and might cause problems for large registries. Use this limit with care.

The custom registry implementations are expected to support at least the wildcard search (*). For example, a pattern of asterisk (*) returns all the users and a pattern of (b*) returns the users starting with *b*.

The return parameter is an object with a com.ibm.websphere.security.Result type. This object contains two attributes, a java.util.List and a java.lang.boolean attribute. The list contains the users that are returned and the Boolean flag indicates if more users are available in the user registry for the search pattern. This Boolean flag is used to indicate to the client whether more users are available in the registry.

In the `FileRegistrySample.java` sample file, the getUsers method retrieves the required number of users from the user registry and sets them as a list in the Result object. To find out if more users are presented than requested, the sample gets one more user than requested and if it finds the additional user, it sets the Boolean flag to `true`. For pattern matching, the match method in the RegExpSample class is used, which supports wildcard characters such as the asterisk (*) and the question mark (?).

This method is called by the administrative console to add users to roles in the various map-users-to-roles panels. The administrative console uses the Boolean set in the Result object to indicate that more entries matching the pattern are available in the user registry.

In WebSphere Application Server Version 4.*x*, this method specifies to take only the pattern parameter. The return is a list. In WebSphere Application Server Version 5.x or later, this method is changed to take one additional parameter, the limit. Ideally, your implementation changes to take the limit value and limits the users that are returned. The return is changed to return a Result object, which consists of the list and a flag that indicates if more entries exist. When the list returns, use the Result.setList(List) method to set the list in the Result object. If more entries exist than requested in the limit parameter, set the Boolean attribute to `true` in the result object, using the Result.setHasMore method. The default for the Boolean attribute in the result object is `false`.

- Obtain the display name of a user with getUserDisplayName(String).

```
public String getUserDisplayName(String userSecurityName)
      throws EntryNotFoundException,
             CustomRegistryException,
             RemoteException;
```

  The getUserDisplayName method returns a display name for a user, if one exists. The display name is an optional string that describes the user that you can set in some registries. This descriptive name is for the user and does not have to be unique in the registry.

  For example, in IBM i systems, you can display the text description for the user profile.

  If you do not need display names in your registry, return null or an empty string for this method.

  If display names existed for any user in WebSphere Application Server Version 4.x, these names were useful for the Enterprise JavaBeans (EJB) method call getCallerPrincipal and the servlet calls getUserPrincipal and getRemoteUser. If the display names are not the same as the security name for any user, the display names are returned for the previously mentioned enterprise beans and servlet methods. Returning display names for these methods might become problematic in some situations because the display names might not be unique in the user registry. Avoid this problem by changing the default behavior to return the user security name instead of the user display name in this version of the product. For more information on how to set properties for the custom registry, see the section on *Setting Properties for Custom Registries*.

  In the `FileRegistrySample.java` sample file, this method returns the display name of the user whose name matches the user name that is provided. If the display name does not exist, this method returns an empty string.

  This method can be called by the product to present the display names in the administrative console, or by using the command line and the **wsadmin** tool. Use this method for display purposes only. This method is the same as in Version 4.x.

- Obtain the unique ID of a user with getUniqueUserId(String).

```
public String getUniqueUserId(String userSecurityName)
       throws EntryNotFoundException,
              CustomRegistryException,
              RemoteException;
```

  This method returns the unique ID of the user, given the security name.

  In the `FileRegistrySample.java` sample file, this method returns the uniqueUserId value of the user whose name matches the supplied name. This method is called when forming a credential for a user and also when creating the authorization table for the application.

- Obtain the security name of a user with getUserSecurityName(String).

```
public String getUserSecurityName(String uniqueUserId)
      throws EntryNotFoundException,
             CustomRegistryException,
             RemoteException;
```

  This method returns the security name of a user given the unique ID. In the `FileRegistrySample.java` sample file, this method returns the security name of the user whose unique ID matches the supplied ID.

  This method is called to make sure a valid user exists for a given uniqueUserId. This method is called to get the security name of the user when the uniqueUserId is obtained from a token.

- Check whether a given user is a valid user in the registry with isValidUser(String).

```
public boolean isValidUser(String userSecurityName)
       throws CustomRegistryException,
              RemoteException;
```

  This method indicates whether the given user is a valid user in the registry.

  In the `FileRegistrySample.java` sample file, this method returns `true` if the user is found in the registry, otherwise this method returns `false`. This method is primarily called in situations where knowing if the

user exists in the directory prevents problems later. For example, in the mapCertificate call, when the name is obtained from the certificate if the user is not found as a valid user in the user registry, you can avoid trying to create the credential for the user.

- Return the list of groups from the user registry with getGroups(String,int).

```
public Result getGroups(String pattern, int limit)
      throws CustomRegistryException,
             RemoteException;
```

The getGroups method returns the list of groups from the user registry. The names of groups depend on the pattern parameter. The number of groups is limited by the limit parameter. In a registry that has many groups, getting all the groups is not practical. So, the limit parameter is introduced to limit the number of groups retrieved from the user registry. A limit of zero (0) implies to return all the groups that match the pattern and can cause problems for large user registries. Use this limit with care. The custom registry implementations are expected to support at least the wildcard search (*). For example, a pattern of asterisk (*) returns all the users and a pattern of (b*) returns the users starting with *b*.

The return parameter is an object of the com.ibm.websphere.security.Result type. This object contains the `java.util.List` and `java.lang.boolean` attributes. The list contains the groups that are returned and the Boolean flag indicates whether more groups are available in the user registry for the pattern searched. This Boolean flag is used to indicate to the client if more groups are available in the registry.

In the `FileRegistrySample.java` sample file, the getUsers method retrieves the required number of groups from the user registry and sets them as a list in the Result object. To find out if more groups are presented than requested, the sample gets one more user than requested and if it finds the additional user, it sets the Boolean flag to `true`. For pattern matching, the match method in the RegExpSample class is used, which supports the asterisk (*) and question mark (?) characters.

This method is called by the administrative console to add groups to roles in the various map-groups-to-roles panels. The administrative console uses the boolean set in the Result object to indicate that more entries matching the pattern are available in the user registry.

In WebSphere Application Server Version 4, this method is used to take the pattern parameter only and returns a list. In WebSphere Application Server Version 5.x or later, this method is changed to take the limit parameter. Change to take the limit value and limit the users that are returned. The return is changed to return a Result object, which consists of the list and a flag that indicates whether more entries exist. Use the Result.setList(List) method to set the list in the Result object. If more entries exist than requested in the limit parameter, set the Boolean attribute to `true` in the Result object using the Result.setHasMore method. The default for the Boolean attribute in the Result object is `false`.

- Obtain the display name of a group with getGroupDisplayName(String).

```
public String getGroupDisplayName(String groupSecurityName)
      throws EntryNotFoundException,
             CustomRegistryException,
             RemoteException;
```

The getGroupDisplayName method returns a display name for a group if one exists. The display name is an optional string that describes the group that you can set in some user registries. This name is a descriptive name for the group and does not have to be unique in the registry. If you do not need to have display names for groups in your registry, return null or an empty string for this method.

In the `FileRegistrySample.java` sample file, this method returns the display name of the group whose name matches the group name that is provided. If the display name does not exist, this method returns an empty string.

The product can call this method to present the display names in the administrative console or through the command line using the **wsadmin** tool. This method is used for display purposes only.

- Obtain the unique ID of a group with getUniqueGroupId(String).

```
public String getUniqueGroupId(String groupSecurityName)
        throws EntryNotFoundException,
               CustomRegistryException,
               RemoteException;
```

This method returns the unique ID of the group that is given the security name.

In the FileRegistrySample.java sample file, this method returns the security name of the group whose unique ID matches the supplied ID. This method verifies that a valid group exists for a given uniqueGroupId ID.

- Obtain the unique IDs of all groups to which a user belongs with getUniqueGroupIds(String).

```
public List getUniqueGroupIds(String uniqueUserId)
      throws EntryNotFoundException,
             CustomRegistryException,
             RemoteException;
```

This method returns the unique IDs of all the groups to which a user belongs.

In the FileRegistrySample.java sample file, this method returns the unique ID of all the groups that contain this uniqueUserID ID. This method is called when creating the credential for the user. As part of creating the credential, all the groupUniqueIds IDs in which the user belongs are collected and put in the credential for authorization purposes when groups are given access to a resource.

- Obtain the security name of a group with getGroupSecurityName(String).

```
public String getGroupSecurityName(String uniqueGroupId)
     throws EntryNotFoundException,
            CustomRegistryException,
            RemoteException;
```

This method returns the security name of a group given its unique ID.

In the FileRegistrySample.java sample file, this method returns the security name of the group whose unique ID matches the supplied ID. This method verifies that a valid group exists for a given uniqueGroupId ID.

- Determine whether a group is a valid group in the registry with isValidGroup(String).

```
public boolean isValidGroup(String groupSecurityName)
       throws CustomRegistryException,
              RemoteException;
```

This method indicates if the given group is a valid group in the registry.

In the FileRegistrySample.java sample file, this method returns true if the group is found in the registry, otherwise the method returns false. This method can be used in situations where knowing whether the group exists in the directory might prevent problems later.

- Obtain all groups to which a user belongs with getGroupsForUser(String).

```
public List getGroupsForUser(String userSecurityName)
     throws EntryNotFoundException,
            CustomRegistryException,
            RemoteException;
```

This method returns all the groups to which a user belongs whose name matches the supplied name. This method is similar to the getUniqueGroupIds method with the exception that the security names are used instead of the unique IDs.

In the FileRegistrySample.java sample file, this method returns all the group security names that contain the userSecurityName name.

This method is called by the administrative console or the scripting tool to verify that the users entered for the RunAs roles are already part of that role in the users and groups-to-role mapping. This check is required to ensure that a user cannot be added to a RunAs role unless that user is assigned to the role in the users and groups-to-role mapping either directly or indirectly through a group that contains this user. Because a group in which the user belongs can be part of the role in the users and groups-to-role mapping, this method is called to check if any of the groups that this user belongs to mapped to that role.

- Retrieve users from a specified group with getUsersForGroup(String,int).

```
public Result getUsersForGroup(String groupSecurityName, int limit)
     throws NotImplementedException,
            EntryNotFoundException,
            CustomRegistryException,
            RemoteException;
```

This method retrieves users from the specified group. The number of users returned is limited by the limit parameter. A limit of zero (0) indicates to return all of the users in that group. This method is not directly called by the WebSphere Application Server security component. However, this method can be called by other components. In rare situations, if you are working with a user registry where getting all the users from any of your groups is not practical, you can create the NotImplementedException exception for the particular groups. In this case, verify that if the process choreographer is installed the staff assignments are not modeled using these particular groups. If no concern exists about returning the users from groups in the user registry, it is recommended that you do not create the NotImplemented exception when implementing this method.

The return parameter is an object with a com.ibm.websphere.security.Result type. This object contains the java.util.List and java.lang.boolean attributes. The list contains the users that are returned and the Boolean flag, which indicates whether more users are available in the user registry for the search pattern. This Boolean flag indicates to the client whether users are available in the user registry.

In the example, this method gets one user more than the requested number of users for a group, if the limit parameter is not set to zero (0). If the method succeeds in getting one more user, the Boolean flag is set to `true`.

In WebSphere Application Server Version 4, this getUsers method is mandatory for the product. For WebSphere Application Server Version 5.x or later, this method can create the `NotImplementedException` exception in situations where it is not practical to get the requested set of users. However, create this exception in rare situations when as other components can be affected. In Version 4, this method accepts only the pattern parameter and returns a list. In Version 5, this method accepts the limit parameter. Change your implementation to take the limit value and limit the users that are returned. The return changes to return a Result object, which consists of the list and a flag that indicates whether more entries exist. When the list is returned, use the Result.setList(List) method to set the list in the Result object. If more entries than requested are in the limit parameter, set the Boolean attribute to `true` in the Result object using Result.setHasMore method. The default for the Boolean attribute in the Result object is `false`.

- Implement the createCredential(String) method.

  Attention: The first two lines of the following code sample are split for illustrative purposes only.

```
public com.ibm.websphere.security.cred.WSCredential createCredential(String userSecurityName)
 throws NotImplementedException,
           EntryNotFoundException,
           CustomRegistryException,
           RemoteException;
```

  In this release the WebSphere Application Server, the createCredential method is not called. You can return *null*. In the example, a *null* value is returned.

What to do next

Managing the realm in a federated repository configuration

Follow this topic to manage the realm in a federated repository configuration.

Before you begin

The realm can consist of identities in:

- The file-based repository that is built into the system
- One or more external repositories
- Both the built-in, file-based repository and in one or more external repositories

Before you configure your realm, review “Federated repositories limitations” on page 1322.

Procedure

1. Configure your realm by using one of the following topics. You might be configuring your realm for the first time or changing an existing realm configuration.

- "Using a single built-in, file-based repository in a new configuration under Federated repositories" on page 1325
- "Changing a federated repository configuration to include a single built-in, file-based repository only" on page 1330
- "Configuring a single, Lightweight Directory Access Protocol repository in a new configuration under Federated repositories" on page 1331
- "Changing a federated repository configuration to include a single, Lightweight Directory Access Protocol repository only" on page 1333
- "Configuring multiple Lightweight Directory Access Protocol repositories in a federated repository configuration" on page 1334
- "Configuring a single built-in, file-based repository and one or more Lightweight Directory Access Protocol repositories in a federated repository configuration" on page 1335

2. Configure supported entity types using the steps described in "Configuring supported entity types in a federated repository configuration" on page 1377. You must configure supported entity types before you can manage this account with Users and Groups. The Base entry for the default parent determines the repository location where entities of the specified type are placed on a create operation.
3. Optional: Use one or more of the following tasks to extend the capabilities of storing data and attributes in your realm:
   a. Configure an entry mapping repository using the steps described in "Configuring an entry mapping repository in a federated repository configuration" on page 1374. An entry mapping repository is used to store data for managing profiles on multiple repositories.
   b. Configure a property extension repository using the steps described in "Configuring a property extension repository in a federated repository configuration" on page 1353. A property extension repository is used to store attributes that cannot be stored in your Lightweight Directory Access Protocol (LDAP) server.
   a. Set up a database repository using wsadmin commands as described in "Setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands" on page 1359
4. Optional: Use one or more of the following advanced user tasks to extend the capabilities of LDAP repositories in your realm:
   - "Increasing the performance of the federated repository configuration" on page 1382
   - "Configuring Lightweight Directory Access Protocol entity types in a federated repository configuration" on page 1396
   - "Configuring group attribute definition settings in a federated repository configuration" on page 1398
5. Optional: Manage repositories that are configured in your system by following the steps described in "Managing repositories in a federated repository configuration" on page 1380.
6. Optional: Add an external repository into your realm by following the steps described in "Adding an external repository in a federated repository configuration" on page 1352.
7. Optional: Change the password for the repository that is configured under federated repositories by the following steps described in "Changing the password for a repository under a federated repositories configuration" on page 1324.

What to do next

1. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.

2. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
3. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Federated repositories:

Federated repositories enable you to use multiple repositories with WebSphere Application Server. These repositories, which can be file-based repositories, LDAP repositories, or a sub-tree of an LDAP repository, are defined and theoretically combined under a single realm. All of the user repositories that are configured under the federated repository functionality are invisible to WebSphere Application Server.

When you use the federated repositories functionality, all of the configured repositories, which you specify as part of the federated repository configuration, become active. It is required that the user ID, and the distinguished name (DN) for an LDAP repository, be unique in multiple user repositories that are configured under the same federated repository configuration. For example, there might be three different repositories that are configured for the federated repositories configuration: Repository A, Repository B, and Repository C. When user1 logs in, the federated repository adapter searches each of the repositories for all of the occurrences of that user. If multiple instances of that user are found in the combined repositories, an error message displays.

In addition, the federated repositories functionality in WebSphere Application Server supports the logical joining of entries across multiple user repositories when the Application Server searches and retrieves entries from the repositories. For example, when an application calls for a sorted list of people whose age is greater than twenty, WebSphere Application searches all of the repositories in the federated repositories configuration. The results are combined and sorted before the Application Server returns the results to the application.

Unlike the local operating system, stand-alone LDAP registry, or custom registry options, federated repositories provide user and group management with read and write capabilities. When you configure federated repositories, you can use one of the following methods to add, create, and delete users and groups:

Important: If you configure multiple repositories under the federated repositories realm, you must also configure supported entity types and specify a base entry for the default parent. The base entry for the default parent determines the repository location where entities of the specified type are placed on write operations by user and group management. See “Configuring supported entity types in a federated repository configuration” on page 1377 for details.

- Use the user management application programming interfaces (API). For more information, refer to articles under "Developing with virtual member manager" in this information center.
- Use the administrative console. To manage users and groups within the administrative console, click **Users and Groups** > **Manage Users** or **Users and Groups** > **Manage Groups**. For information on user and group management, click the Help link that displays in the upper right corner of the window. From the left navigation pane, click **Users and Groups**.
- Use the wsadmin commands. For more information, see the WIMManagementCommands command group for the AdminTask object topic.

If you do not configure the federated repositories functionality or do not enable federated repositories as the active repository, you cannot use the user management capabilities that are associated with federated repositories. You can configure an LDAP server as the active user registry and configure the same LDAP server under federated repositories, but not select federated repositories as the active user repository. With this scenario, authentication takes place using the LDAP server, and you can use the user management functionality for the LDAP server that is available for federated repositories.

The following table compares the federated repository functionality that is available in WebSphere Application Server Version 8.0 with the registry functionality that remains unchanged from previous versions of the Application Server.

Table 84. Federated repositories versus user registry implementations.

This table lists federated repositories versus user registry implementations.

Federated repositories	User registry
Supports multiple types of repositories such as file-based, LDAP, database, and custom. In WebSphere Application Server Version 8.0, file-based and LDAP repositories are supported by the administrative console. However, the federated repositories functionality does not support local operating system implementations. With this service release, the federated repositories functionality supports local operating system implementations. For database and custom repositories, you can use the wsadmin command-line interface or the configuration application programming interfaces (API).	Supports multiple types of registries such as the local operating system, a stand-alone LDAP registry, and a stand-alone custom registry.
Supports multiple repositories in a realm within a cell.	Supports one registry only in a realm within a cell.
Provides read and write capabilities for the repositories that are defined in the federated repository configuration.	Provides read only capability for the registries.
Provides account and password policy support as defined by the registry type. However, this support is not provided by the federated repository functionality.	Provides account and password policy support as defined by the registry type.
Supports identity profiles.	Does not support identity profiles.
Uses the custom UserRegistry implementation.	Uses the custom UserRegistry implementation.

Realm configuration settings:

Use this page to manage the realm. The realm can consist of identities in the file-based repository that is built into the system, in one or more external repositories, or in both the built-in, file-based repository and one or more external repositories.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Security domains**.
2. Under User realm, select **Customize for this domain**. Select **Federated repositories** from the Realm type field and click **Configure**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

A single built-in, file-based repository is built into the system and included in the realm by default.

You can configure one or more Lightweight Directory Access Protocol (LDAP) repositories to store identities in the realm. Click **Add base entry to realm** to specify a repository configuration and a base entry into the realm. You can configure multiple different base entries into the same repository.

Click **Remove** to remove selected repositories from the realm. Repository configurations and contents are not destroyed. The following restrictions apply:

- The realm must always contain at least one base entry; therefore, you cannot remove every entry.
- If you plan to remove the built-in, file-based repository from the administrative realm, verify that at least one user in another member repository is a console user with administrative rights. Otherwise, you must disable security to regain access to the administrative console.

WebSphere Application Server Version 7.0 distinguishes between the user identities for administrators who manage the environment and server identities for authenticating server to server communications. In most cases, server identities are automatically generated and are not stored in a repository.

Realm name:

Specifies the name of the realm. You can change the realm name.

Primary administrative user name:

Specifies the name of the user with administrative privileges that is defined in the repository, for example, adminUser.

The user name is used to log on to the administrative console when administrative security is enabled. Version 6.1 requires an administrative user that is distinct from the server user identity so that administrative actions can be audited.

Attention: In WebSphere Application Server, Version 6.0.x, a single user identity is required for both administrative access and internal process communication. When migrating to Version 6.1, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Automatically generated server identity:

Enables the application server to generate the server identity, which is recommended for environments that contain only Version 6.1 or later nodes. Automatically generated server identities are not stored in a user repository.

Default: Enabled

Server identity that is stored in the repository:

Specifies a user identity in the repository that is used for internal process communication. Cells that contain Version 6.1 or later nodes require a server user identity that is defined in the active user repository.

Default: Enabled

Ignore case for authorization:

Specifies that a case-insensitive authorization check is performed.

If case sensitivity is not a consideration for authorization, enable the **Ignore case for authorization** option.

Allow operations if some of the repositories are down:

Specifies whether operations (such as login, search, or get) are allowed even if the repositories in the realm are down.

Use global schema for model:

Sets the global schema option for the data model in a multiple security domain environment. Global schema refers to the schema of the admin domain.

Note: Application domains that are set to use global schema share the same schema of the admin domain. If you extend the schema for an application in one domain, you must also consider how that might affect applications of other domains, as they are bound by the same schema. For example, adding a mandatory property for one application might cause other applications to fail.

Base entry:

Specifies the base entry within the realm. This entry and its descendents are part of the realm.

Repository identifier:

Specifies a unique identifier for the repository. This identifier uniquely identifies the repository within the cell.

Repository type:

Specifies the repository type, such as File or LDAP.

User attribute mapping for federated repositories:

Use this panel to set or to modify the mapping for a user registry's user or group attribute to a federated repository property.

To view this administrative console page, click **Security > Global security**. Under Available realm definitions, click **Federated repositories**, and then **Configure**. On the next panel and under Additional Properties, click **User repository attribute mapping**.

Attribute mappings:

Select an attribute to set or to modify the mapping for a user registry's user or group attribute to a federated repository property, and then click **Edit**.

Custom properties details for federated repositories:

Use this panel to specify the configuration for access to a custom repository.

To view this administrative console page, click **Security > Global security**. Under Available realm definitions, click **Federated repositories**, and then **Configure**. On the next panel and under Additional Properties, click **Manage repositories**. Under Add, select **Custom repository**.

Repository identifier:

Specifies a unique identifier for the repository. This identifier uniquely identifies the repository within the cell.

Repository adapter class name:

Specifies the implementation class name for the repository adapter. For a User Registry bridge, use com.ibm.ws.wim.adapter.urbridge.URBridge.

Login properties:

Specifies the property names to use to log into the application server.

Custom properties:

Specifies arbitrary name and value pairs of data. The name is a property key and the value is a string value that can be used to set internal system configuration properties.

File details for federated repositories:

Use this panel to specify the configuration for access to a file repository.

To view this administrative console page, click **Security > Global security**. Under Available realm definitions, click **Federated repositories**, and then **Configure**. On the next panel and under Additional Properties, click **Manage repositories**. Under Add, select **File repository**.

Repository identifier:

Specifies a unique identifier for the repository. This identifier uniquely identifies the repository within the cell.

Repository adapter class name:

Specifies the implementation class name for the repository adapter. For a User Registry bridge, use com.ibm.ws.wim.adapter.urbridge.URBridge.

Base directory:

The base directory where the files are to be created. This directory must already exist.

File name:

The file name for the repository.

The default value is `fileRegistry.xml`.

Salt length:

Specifies the salt length of the randomly generated salt for password hashing.

The default value is 12.

Message digest algorithm:

Specifies the message digest algorithm to use for hashing the password.

Select one of the following: SHA-1, SHA-256, SHA-384 or SHA-512.

The default value is SHA-1.

Login properties:

Specifies the property names to use to log into the application server.

Custom properties:

Specifies arbitrary name and value pairs of data. The name is a property key and the value is a string value that can be used to set internal system configuration properties.

Federated repositories limitations:

This topic outlines known limitations and important information for configuring federated repositories.

Configuring federated repositories in a mixed-version environment

In a mixed-version deployment manager cell that contains both Version 6.1.x and Version 5.x or 6.0.x nodes, the following limitations apply for configuring federated repositories:

- You can configure only one Lightweight Directory Access Protocol (LDAP) repository under federated repositories, and the repository must be supported by Version 5.x or 6.0.x.
- You can specify a realm name that is compatible with prior versions only. The host name and the port number represent the realm for the LDAP server in a mixed-version nodes cell. For example, machine1.austin.ibm.com:389.
- You must configure a stand-alone LDAP registry; the LDAP information in both the stand-alone LDAP registry and the LDAP repository under the federated repositories configuration must match. During node synchronization, the LDAP information from the stand-alone LDAP registry propagates to the Version 5.x or 6.0.x nodes.

  Important: Before node synchronization, verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. Do not set the stand-alone LDAP registry as the current realm definition.
- You cannot configure an entry mapping repository or a property extension repository in a mixed-version deployment manager cell.

Configuring LDAP servers in a federated repository

The LDAP connection `connectTimeout` default value is 20 seconds. LDAP should respond within 20 seconds for any request from WebSphere Application Server. If you cannot connect to your LDAP within this time, make sure that your LDAP is running. A connection error displays at the top of the LDAP configuration panel when the connection timeout exceeds 20 seconds.

Coexisting with Tivoli Access Manager

For Tivoli Access Manager to coexist with a federated repositories configuration, the following limitations apply:

- You can configure only one LDAP repository under federated repositories, and that LDAP repository configuration must match the LDAP server configuration under Tivoli Access Manager.
- The distinguished name for the realm base entry must match the LDAP distinguished name (DN) of the base entry within the repository. In WebSphere Application Server, Tivoli Access Manager recognizes the LDAP user ID and LDAP DN for both authentication and authorization. The federated repositories configuration does not include additional mappings for the LDAP user ID and DN.
- The federated repositories functionality does not recognize the metadata that is specified by Tivoli Access Manager. When users and groups are created under user and group management, they are not formatted using the Tivoli Access Manager metadata. The users and groups must be manually imported into Tivoli Access Manager before you use them for authentication and authorization.

Limitation for configuring active directories with their own federated repository realms

In order to use the administrative console to perform a wildcard search for all available users on two Active Directories, and to prevent multiple entries exceptions with all built-in IDs, you must first configure each Active Directory with it's own federated repository realm.

However, you cannot use the administrative console to configure each Active Directory with it's own federated repository realm. You can instead use a wsadmin script similar to the following:

```
$AdminTask createIdMgrRealm {-name AD1realm}
$AdminTask addIdMgrRealmBaseEntry {-name AD1realm -baseEntry o=AD1}

$AdminTask createIdMgrRealm {-name AD2realm}
$AdminTask addIdMgrRealmBaseEntry {-name AD2realm -baseEntry o=AD2}

$AdminConfig save
```

Changing the password for a repository under a federated repositories configuration:

Passwords allow security control over the repositories under a federated repositories configuration. As part of managing the realm in a federated repository configuration, one of the optional tasks you can perform is to change the password of an individual repository that is under a federated repositories configuration.

Before you begin

Before you change the password for the repository that is configured under federated repositories, ensure that the WebSphere Application Server is running and the target repository for the password change is configured under the federated repositories configuration.

Procedure

- Changing the password for a repository using the dynamic **updateIdMgrLDAPBindInfo** command Use the following steps to change the Lightweight Directory Access Protocol (LDAP) bind distinguished name (DN) or bind password of an LDAP repository.

  From a **wsadmin** prompt, you can enter the following command to display a list of arguments for the **updateIdMgrLDAPBindInfo** command: `$AdminTask help updateIdMgrLDAPBindInfo`

  1. Start the wsadmin command-line utility. The wsadmin command is found in the*app_server_root*`/bin` directory. The WebSphere Application Server and wsadmin must remaining running.
  2. Use an LDAP tool to change the password of the LDAP repository. Some LDAP repositories require a stop and start of the LDAP server to change the password.
  3. From the **wsadmin** prompt, enter the **updateIdMgrLDAPBindInfo** command to update the LDAP password under the federated repository. The change is also reflected in the wimconfig.xml file.
- Changing the password for a repository using the **updateIdMgrDBRepository** command
  1. Start the wsadmin command-line utility. The wsadmin command is found in the *app_server_root*`/bin` directory. The wsadmin command session must remain running. If WebSphere Application Server is not started, you need to open a wsadmin command session in local mode. `wsadmin -conntype none`

     gotcha: If you are starting the wsadmin command session in local mode, you must ensure that the location of the database driver is specified in the class path using the –wsadmin_classpath option. For information on using this option, see the topic, *wsadmin scripting tool* in the WebSphere Application Server information center.
  2. Log in to the Administrative Console for WebSphere Application Server.
  3. Change the password for the repository.
  4. From the Administrative Console, change the data source (J2C) password. You access the proper console page by clicking **Resources > JDBC > Data sources >***data_source***> JAAS - J2C authentication data**.
  5. From the Administrative Console, save your changes to the master configuration.
  6. From the **wsadmin** prompt, use the updateIdMgrDBRepository command to update the password in the wimconfig.xml file.
  7. From the **wsadmin** prompt, save your changes to the master configuration. The following command is used to save the master configuration: `$AdminConfig save`.
  8. Restart the WebSphere Application Server.
- Changing the password for a repository using the **setIdMgrPropertyExtensionRepository** command
  1. Start the wsadmin command-line utility. The wsadmin command is found in the*app_server_root*`/bin` directory. The wsadmin command session must remain running. If WebSphere Application Server is not started, you need to open a wsadmin command session in local mode.

     ```
     wsadmin -conntype none
     ```
  2. Log in to the Administrative Console for WebSphere Application Server.
  3. Change the password for the repository.

4. From the Administrative Console, change the data source (J2C) password. You access the proper console page by clicking **Resources > JDBC > Data sources >** *data_source* **> JAAS - J2C authentication data**.
5. From the Administrative Console, save your changes to the master configuration.
6. From the **wsadmin** prompt, use the **setIdMgrPropertyExtensionRepository** command to update the password in the wimconfig.xml file.
7. From the **wsadmin** prompt, save your changes to the master configuration. The following command is used to save the master configuration: `$AdminConfig save`.
8. Restart the WebSphere Application Server.

- Changing the password for a repository using the **setIdMgrEntryMappingRepository** command
  1. Start the wsadmin command-line utility. The wsadmin command is found in the*app_server_root*`/bin` directory. The wsadmin command session must remain running. If WebSphere Application Server is not started, you need to open a wsadmin command session in local mode.

     ```
     wsadmin -conntype none
     ```
  2. Log in to the Administrative Console for WebSphere Application Server.
  3. Change the password for the repository.
  4. From the Administrative Console, change the data source (J2C) password. You access the proper console page by clicking **Resources > JDBC > Data sources >***data_source* **> JAAS - J2C authentication data**.
  5. From the Administrative Console, save your changes to the master configuration.
  6. From the **wsadmin** prompt, use the **setIdMgrEntryMappingRepository** command to update the password in the wimconfig.xml file.
  7. From the **wsadmin** prompt, save your changes to the master configuration. The following command is used to save the master configuration: `$AdminConfig save`.
  8. Restart the WebSphere Application Server.
- Changing the password for a repository using the **updateIdMgrLDAPServer** command
  1. Start the wsadmin command-line utility. The wsadmin command is found in the*app_server_root*`/bin` directory. The wsadmin command session must remain running. If WebSphere Application Server is not started, you need to open a wsadmin command session in local mode.

     ```
     wsadmin -conntype none
     ```
  2. Change the password for the repository.
  3. From the **wsadmin** prompt, use the **updateIdMgrLDAPServer** command to update the password in the wimconfig.xml file.
  4. From the **wsadmin** prompt, save your changes to the master configuration. The following command is used to save the master configuration: `$AdminConfig save`.
  5. Restart the WebSphere Application Server.

Results

The password for the repository has been changed.

Using a single built-in, file-based repository in a new configuration under Federated repositories:

Follow this task to use a single built-in, file-based repository in a new configuration under Federated repositories.

Before you begin

To use the default configuration under Federated repositories that includes a single built-in, file-based repository only, you need to know the primary administrative user name of the user who manages WebSphere Application Server resources and user accounts.

Restriction:

Client certificate login is not supported in a realm that includes a single built-in, file-based repository or a single built-in, file-based repository with other repositories.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Leave the Realm name field value as `defaultWIMFileBasedRealm`.
4. Enter the name of the primary administrative user in the Primary administrative user name field, for example, adminUser.
5. Leave the **Ignore case for authorization** option enabled.
6. Click **OK**.
7. Provide an administrative user password. This panel displays only when a built-in, file-based repository is included in the realm. Otherwise, the panel does not display. If a built-in, file-based repository is included, complete the following steps:
   a. Supply a password for the administrative user in the Password field.
   b. Confirm the password of the primary administrative user in the Confirm password field.
   c. Click **OK**.

Results

After completing these steps, your new configuration under Federated repositories includes a single built-in, file-based repository only.

What to do next

1. Before you can manage this account with Users and Groups, configure supported entity types as described in “Configuring supported entity types in a federated repository configuration” on page 1377.
2. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
3. If you are enabling security, complete the remaining steps, as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
4. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Administrative user password settings:

Use this page to set a password for the administrative user who manages the product resources and user accounts.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. If your federated repository configuration includes a built-in, file-based repository, then the **Administrative user password** panel displays when changes are applied.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Password:

Specifies the password of the administrative user who manages the product resources and user accounts.

Confirm password:

Confirms the password of the administrative user who manages the product resources and user accounts.

Federated repository wizard settings:

Use this security wizard page to complete the basic requirements to connect the application server to a federated repository.

To view this security wizard page, complete the following steps

1. Click **Security > Global security > Security configuration wizard**.
2. Select your protection settings and click **Next**.
3. Select the **Federated repositories** option and click **Next**.

You can modify your federated repository configuration by completing the following steps:

1. Click **Security > Global security**.
2. Under User account repository, select Federated repository and click **Configure**.

Note: This wizard is used for the initial configuration of a built-in, file-based repository. The user name and password do not have to be in the federated repository because they will be created. If you have previously configured federated repositories, do not use the Security configuration wizard to modify your configuration. Instead, modify your configuration using the Federated repositories selection under User account repository on the Global security panel.

Primary administrative user name:

Specifies the name of the user with administrative privileges that is defined in the repository, for example, adminUser.

The user name is used to log on to the administrative console when administrative security is enabled. Version 6.1 requires an administrative user that is distinct from the server user identity so that administrative actions can be audited.

Attention: In WebSphere Application Server, Version 6.0.x, a single user identity is required for both administrative access and internal process communication. When migrating to Version 6.1, this identity is used as the server user identity. You need to specify another user for the administrative user identity.

Password:

Specifies the password of the administrative user who manages the product resources and user accounts.

Confirm password:

Confirms the password of the administrative user who manages the product resources and user accounts.

Configuring a single built-in, file-based repository in a new configuration under federated repositories using wsadmin:

You can use the Jython or Jacl scripting language with the wsadmin tool to configure a single built-in, file-based repository in a new configuration under Federated repositories.

Before you begin

Shut down the application server and ensure you have the primary administrator id and password.

About this task

The federated repositories configuration file, `wimconfig.xml`, is supported by WebSphere Application Server 6.1.x and is located in the *app_server_root*/`profiles/profile_name/config/cells/cell_name/wim/config` directory.

Use the following steps to configure for use a single built-in, file-based repository in a new configuration for federated repositories.

Procedure

1. Start the wsadmin scripting tool.
2. Create the `fileRegistry.xml` file, which is the user registry itself, if it does not already exist. If the `fileRegistry.xml` file does exist, this step just adds the user to registry.

   Using Jython:

   ```
   AdminTask.addFileRegistryAccount('-userId isoet01s01 -password oets01')
   ```

   Using Jacl:

   ```
   $AdminTask addFileRegistryAccount {-userId isoet01s01 -password oets01}
   ```

   For more information on the addFileRegistryAccount command, see the documentation about the FileRegistryCommands command group for the AdminTask object.
3. Update the `security.xml` file to enable administrative security, set the activeUserRegistry to use federated repositories, and update the primaryAdmin and its password.

   Using Jython:

   ```
   AdminTask applyWizardSettings('-secureApps false
    -secureLocalResources false
    -userRegistryType WIMUserRegistry
    -customRegistryClass com.ibm.ws.wim.registry.WIMUserRegistry
    -adminName isoet01s01 -adminPassword oets01')
   ```

   Using Jacl:

   ```
   $AdminTask applyWizardSettings {-secureApps false
    -secureLocalResources false
    -userRegistryType WIMUserRegistry
    -customRegistryClass com.ibm.ws.wim.registry.WIMUserRegistry
    -adminName isoet01s01
    -adminPassword oets01}
   ```

   For more information on the applyWizardSettings command, see the documentation about the WizardCommands command group for the AdminTask object.
4. Save your configuration changes. Enter the following commands to save the new configuration and close the wsadmin tool:

   Using Jython:

   ```
   AdminConfig.save()
   ```

   Using Jacl:

   ```
   $AdminConfig save
   ```
5. Restart the application server.

FileRegistryCommands command group for the AdminTask object:

Federated repositories provides a file registry. Use the commands in the FileRegistryCommands command group to administer the file registry using the wsadmin tool.

Note: If the Use global security settings option is selected for the user realm or the Global federated repositories option is selected as the realm type for the specified domain, the user and group management commands are executed on the federated repository of the admin domain. For example, if you run the createUser command for the specified domain, the user is created in the admin domain. However, configuration changes that are performed on the domain are applied to the security domain-specific configuration.

Use the following commands in the FileRegistryCommands group to modify the federated repository file registry:

- "addFileRegistryAccount command"
- "changeFileRegistryAccountPassword command" on page 1330

addFileRegistryAccount command

The addFileRegistryAccount command adds an account to the file registry. You must save your configuration changes after running this command to save the new account to the master repository.

Target object

None

Required parameters

-userId
Specifies the ID of the user to add to the file registry. (String, required)

-password
Specifies the password of the user to add to the file registry. (String, required)

Optional parameters

-securityDomainName
Specifies the name that uniquely identifies the security domain. If you do not specify this parameter, the command uses the global federated repository. (String, optional)

-parent
Specifies the parent of the entity. (String, optional)

.

Return value

This command returns a message that indicates that the command ran successfully, as the following example displays:

```
'CWWIM4544I Account newAcct(uid=newAcct,o=defaultWIMFileBasedRealm) is stored in the
file registry in the temporary workspace. You must use the "$AdminConfig save"
command to save it in the master repository.'
```

Batch mode example usage

- Using Jython string:

```
AdminTask.addFileRegistryAccount('[-userId newAcct -password new22password]')
```

Interactive mode example usage

- Using Jython string:

```
AdminTask.addFileRegistryAccount(['-userId', 'newAcct', '-password', 'new22password'])
```

changeFileRegistryAccountPassword command

The changeFileRegistryAccountPassword changes the password for the file registry account.

Target object

None.

Required parameters

-userId
Specifies the user ID of interest. (String, required)

-password
Specifies the new password. (String, required)

Optional parameters

-securityDomainName
Specifies the name that uniquely identifies the security domain. If you do not specify this parameter, the command uses the global federated repository. (String, optional)

-uniqueName
Specifies the fully-qualified unique name of the administrator. (String, optional)

Return value

This command returns a message that indicates that the command ran successfully, as the following example displays:

```
'CWWIM4545I The password is changed for newAcct(uid=newAcct,o=defaultWIMFileBasedRealm)
in the file registry in the temporary workspace. You must use the "$AdminConfig save"
command to save it in the master repository.'
```

Batch mode example usage

- Using Jython string:

```
AdminTask.changeFileRegistryAccountPassword('[-userId newAcct -password newPassword -uniqueName
uid=newAcct,o=defaultWIMFileBasedRealm]')
```

Interactive mode example usage

- Using Jython string:

```
AdminTask.changeFileRegistryAccountPassword(['-userId', 'newAcct', '-password', 'newPassword',
'-uniqueName', 'uid=newAcct,o=defaultWIMFileBasedRealm'])
```

Changing a federated repository configuration to include a single built-in, file-based repository only:

Follow this task to change your federated repository configuration to include a single built-in, file-based repository only.

Before you begin

To change your federated repository configuration to include a single built-in, file-based repository only, you need to know the primary administrative user name of the user who manages WebSphere Application Server resources and user accounts.

Restriction: Client certificate login is not supported in a realm that includes a single built-in, file-based repository or a single built-in, file-based repository with other repositories.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Enter the name of the realm in the Realm name field. If the realm contains a single built-in, file-based repository only, you must specify `defaultWIMFileBasedRealm` as the realm name.
4. Enter the name of the primary administrative user in the Primary administrative user name field, for example, adminUser.
5. Enable the **Ignore case for authorization** option.
6. Click **Use built-in repository** if the built-in, file-based repository is not listed in the collection.
7. Select all repositories in the collection that are not of type File and click **Remove**.
8. Click **OK**.
9. Provide an administrative user password. This panel displays only when a built-in, file-based repository is included in the realm. Otherwise, it does not display. If a built-in, file-based repository is included, complete the following steps:
   a. Supply a password for the primary administrative user in the Password field.
   b. Confirm the password of the primary administrative user in the Confirm password field.
   c. Click **OK**.

Results

After completing these steps, your federated repository configuration, which includes a single built-in, file-based repository only, is configured.

What to do next

1. Before you can manage this account with Users and Groups, configure supported entity types as described in “Configuring supported entity types in a federated repository configuration” on page 1377.
2. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
3. If you are enabling security, complete the remaining steps, as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
4. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Configuring a single, Lightweight Directory Access Protocol repository in a new configuration under Federated repositories:

Follow this task to configure a single, Lightweight Directory Access Protocol (LDAP) repository in a new configuration under Federated repositories.

Before you begin

To configure an LDAP repository in a new configuration under Federated repositories, you must know a valid user name (ID), the user password, the server host and port and, if necessary, the bind distinguished name (DN) and the bind password. You can choose any valid user in the repository that is searchable. In some LDAP servers, administrative users are not searchable and cannot be used (for example, cn=root in SecureWay). This user is referred to as the WebSphere Application Server *administrative user name* or

administrative ID in the documentation. Being an administrative ID means a user has special privileges when calling some protected internal methods. Normally, this ID and password are used to log in to the administrative console after you turn on security. You can use other users to log in, if those users are part of the administrative roles.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. On the Federated repositories panel, complete the following steps:
   a. Enter the name of the realm in the Realm name field. You can change the existing realm name.
   b. Enter the name of the primary administrative user in the Primary administrative user name field, for example, adminUser.
   c. Optional: Select the **Ignore case for authorization** option. When you enable this option, the authorization check is case-insensitive. Normally, an authorization check involves checking the complete DN of a user, which is unique in the realm and is case-insensitive. Clear this option when all of the member repositories in the realm are case-sensitive.

      Restriction: Some repositories contain data that is case-sensitive only, and some repositories contain data that is case-insensitive only. Do not include both case-sensitive and case-insensitive repositories in the realm. For example, do not include case-sensitive repositories in the realm with a built-in, file-based repository.
   d. Click **Add base entry to realm** to add a base entry that uniquely identifies the external repository in the realm. Then complete the steps in “Adding an external repository in a federated repository configuration” on page 1352.
4. On the Federated repositories panel, complete the following steps:
   a. Select the built-in, file-based repository in the collection, and click **Remove**.

      Restriction: Before you remove the built-in, file-based repository from the administrative realm, verify that at least one user in another member repository is a console user with administrative rights. Otherwise, you must disable security to regain access to the administrative console.
   b. Click **OK**.

Results

After completing these steps, your new configuration under Federated repositories includes a single, LDAP repository only.

What to do next

1. Before you can manage this account with Users and Groups, configure supported entity types as described in “Configuring supported entity types in a federated repository configuration” on page 1377.
2. After configuring the federated repositories, click **Security > Global secuity** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
3. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.

4. Save, stop, and restart all the product servers (deployment managers, nodes and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Changing a federated repository configuration to include a single, Lightweight Directory Access Protocol repository only:

Follow this task to change your federated repository configuration to include a single, Lightweight Directory Access Protocol repository (LDAP) repository only.

Before you begin

To configure an LDAP repository in a federated repository configuration, you must know a valid user name (ID), the user password, the server host and port and, if necessary, the bind distinguished name (DN) and the bind password. You can choose any valid user in the repository that is searchable. In some LDAP servers, administrative users are not searchable and cannot be used (for example, cn=root in SecureWay). This user is referred to as a WebSphere Application Server *administrative user name* or *administrative ID* in the documentation. Being an administrative ID means a user has special privileges when calling some protected internal methods. Normally, this ID and password are used to log into the administrative console after you turn on security. You can use other users to log in if those users are part of the administrative roles.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Enter the name of the realm in the Realm name field. You can change the existing realm name.
4. Enter the name of the primary administrative user in the Primary administrative user name field, for example, adminUser.
5. Optional: Select the **Ignore case for authorization** option. When you enable this option, the authorization check is case-insensitive. Normally, an authorization check involves checking the complete DN of a user, which is unique in the realm and is case-insensitive. Clear this option when all of the member repositories in the realm are case-sensitive.

   Restriction: Some repositories contain data that is case-sensitive only, and some repositories contain data that is case-insensitive only. Do not include both case-sensitive and case-insensitive repositories in the realm. For example, do not include case-sensitive repositories in the realm with a built-in, file-based repository.
6. Optional: Click **Add base entry to realm** if the LDAP repository that you need is not contained in the collection. Then complete the steps in “Adding an external repository in a federated repository configuration” on page 1352.
7. On the Federated repositories panel, complete the following steps:
   a. Optional: Select the repositories in the collection that you do not need in the realm and click **Remove**.

      Restriction: The realm must always contain at least one base entry; therefore, you cannot remove every entry.
   b. Click **OK**.

Results

After completing these steps, your federated repository configuration, which includes a single LDAP repository only, is configured.

What to do next

1. Before you can manage this account with Users and Groups, configure supported entity types as described in “Configuring supported entity types in a federated repository configuration” on page 1377.
2. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
3. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
4. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Configuring multiple Lightweight Directory Access Protocol repositories in a federated repository configuration:

Follow this task to configure multiple Lightweight Directory Access Protocol (LDAP) repositories in a federated repository configuration.

Before you begin

To configure an LDAP repository in a federated repository configuration, you must know a valid user name (ID), the user password, the server host and port and, if necessary, the bind distinguished name (DN) and the bind password. You can choose any valid user in the repository that is searchable. In some LDAP servers, administrative users are not searchable and cannot be used (for example, cn=root in SecureWay). This user is referred to as a WebSphere Application Server *administrative user name* or *administrative ID* in the documentation. Being an administrative ID means a user has special privileges when calling some protected internal methods. Normally, this ID and password are used to log into the administrative console after you turn on security. You can use other users to log in if those users are part of the administrative roles.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Enter the name of the realm in the Realm name field. You can change the existing realm name.
4. Enter the name of the primary administrative user in the Primary administrative user name field, for example, adminUser.
5. Optional: Select the **Ignore case for authorization** option. When you enable this option, the authorization check is case-insensitive. Normally, an authorization check involves checking the complete DN of a user, which is unique in the realm and is case-insensitive. Clear this option when all of the member repositories in the realm are case-sensitive.

   Restriction: Some repositories contain data that is case-sensitive only, and some repositories contain data that is case-insensitive only. Do not include both case-sensitive and case-insensitive repositories in the realm. For example, do not include case-sensitive repositories in the realm with a built-in, file-based repository.
6. Optional: Click **Add base entry to realm** if the LDAP repository that you need is not listed in the collection. Then complete the steps in “Adding an external repository in a federated repository configuration” on page 1352.
7. On the Federated repositories panel, complete the following steps:

a. Optional: Repeat step 6 if the LDAP repository that you need is not listed in the collection.
b. Optional: Select the repositories in the collection that you do not need in the realm and click **Remove**. The following restrictions apply:
   - The realm must always contain at least one base entry; therefore, you cannot remove every entry.
   - If you plan to remove the built-in, file-based repository from the administrative realm, verify that at least one user in another member repository is a console user with administrative rights. Otherwise, you must disable security to regain access to the administrative console.
c. Click **OK**.

Results

After completing these steps, your federated repository configuration, which includes multiple LDAP repositories, is configured.

What to do next

1. Before you can manage this account with Users and Groups, configure supported entity types as described in "Configuring supported entity types in a federated repository configuration" on page 1377.
2. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
3. If you are enabling security, complete the remaining steps as specified in "Enabling security for the realm" on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
4. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Configuring a single built-in, file-based repository and one or more Lightweight Directory Access Protocol repositories in a federated repository configuration:

Follow this task to configure a single built-in, file-based repository and multiple Lightweight Directory Access Protocol (LDAP) repositories in a federated repository configuration.

Before you begin

To configure a built-in, file-based repository in a federated repository configuration, you must know the primary administrative user name of the user who manages WebSphere Application Server resources and user accounts.

To configure an LDAP repository in a federated repository configuration, you must know a valid user name (ID), the user password, the server host and port and, if necessary, the bind distinguished name (DN) and the bind password. You can choose any valid user in the repository that is searchable. In some LDAP servers, administrative users are not searchable and cannot be used (for example, cn=root in SecureWay). This user is referred to as a WebSphere Application Server *administrative user name* or *administrative ID* in the documentation. Being an administrative ID means a user has special privileges when calling some protected internal methods. Normally, this ID and password are used to log in to the administrative console after you turn on security. You can use other users to log in if those users are part of the administrative roles.

Restriction: Client certificate login is not supported in a realm that includes a single built-in, file-based repository or a single built-in, file-based repository with other repositories.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Enter the name of the realm in the Realm name field. You can change the existing realm name.
4. Enter the name of the primary administrative user in the Primary administrative user name field, for example, adminUser.

   Restriction: When you configure multiple repositories that includes a single built-in, file-based repository, the primary administrative user name must exist in the file-based repository. If the primary administrative user name does not exist in the file-based repository, then the name is created in the file-based repository. The primary administrative user name cannot exist in other repositories.
5. Select the **Ignore case for authorization** option.

   Attention: When the realm includes a built-in, file-based repository, you must enable the **Ignore case for authorization** option.
   When you enable this option, the authorization check is case-insensitive. Normally, an authorization check involves checking the complete DN of a user, which is unique in the realm and is case-insensitive. Clear this option when all of the member repositories in the realm are case-sensitive.

   Restriction: Some repositories contain data that is case-sensitive only, and some repositories contain data that is case-insensitive only. Do not include both case-sensitive and case-insensitive repositories in the realm. For example, do not include case-sensitive repositories in the realm with a built-in, file-based repository.
6. Optional: Click **Add base entry to realm** if the LDAP repository that you need is not contained in the collection. Then complete the steps in “Adding an external repository in a federated repository configuration” on page 1352.
7. On the Federated repositories panel, complete the following steps:
   a. Optional: Repeat step 6 if the LDAP repository that you need is not listed in the collection.
   b. Click **Use built-in repository** if the built-in, file-based repository is not listed in the collection.
   c. Optional: Select the repositories in the collection that you do not need in the realm and click **Remove**.

      Restriction: The realm must always contain at least one base entry; therefore, you cannot remove every entry.
   d. Click **OK**.
8. Provide an administrative user password. This panel displays only when a built-in, file-based repository is included in the realm. Otherwise, the panel does not display. If a built-in, file-based repository is included, complete the following steps:
   a. Supply a password for the administrative user in the Password field.
   b. Confirm the password of the primary administrative user in the Confirm password field.
   c. Click **OK**.

Results

After completing these steps, your federated repository configuration, which includes a single built-in, file-based repository and one or more LDAP repositories, is configured.

What to do next

1. Before you can manage this account with Users and Groups, configure supported entity types as described in “Configuring supported entity types in a federated repository configuration” on page 1377.

2. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
3. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
4. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Manually configuring an Lightweight Directory Access Protocol repository in a federated repository configuration:

Follow this topic to manually configure Lightweight Directory Access Protocol (LDAP) repository in a federated repository configuration.

Before you begin

As a prerequisite, you need to add a LDAP repository to your WebSphere Application Server configuration, where you define the following information:

Table 85. Prerequisite LDAP repository information.

This table lists prerequisite LDAP repository information,

Item Name	Example
Repository identifier	`ldaprepo1`
Directory type	IBM Tivoli Directory Server
Primary host name	`localhost`
Port	`389`
Bind distinguished name	`cn=ldapadmin`
Bind password	`yourpwd`
Login properties	`uid` (a property containing login information)

See “Lightweight Directory Access Protocol repository configuration settings” on page 1343 for the specific steps you must perform to establish this LDAP repository.

About this task

At this point, you have a valid LDAP repository ready to be manually configured in a federated repository configuration.

Procedure

1. Map the federated repository entity types to the LDAP object classes.
   a. Configure the LDAP repository to match the used LDAP object class for users.
      1) In the administrative console, click **Security > Global security**.
      2) Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
      3) Under Related items, click **Manage repositories**.
      4) Select the repository (for example, `ldaprepo1`).
      5) Click **LDAP entity types**.

6) Click **PersonAccount**.
7) Insert the *objectclass* name used in our LDAP server, for example, `inetOrgPerson`.
8) Click **Apply**.
9) Click **Save**.

See “Configuring supported entity types in a federated repository configuration” on page 1377 for an explanation of the supported entity types.

See http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.wim.doc.en/ldap.html for a description of the LDAP default mappings.

b. Configure the LDAP repository to match the used LDAP *objectclass* for groups
1) In the administrative console, click **Security > Global security**.
2) Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3) Under Related items, click **Manage repositories**.
4) Select `ldaprepo1`.
5) Click **LDAP entity types**.
6) Click **Group**.
7) Insert the objectclass name used for your LDAP server, for example, `groupOfUniqueNames`.
8) Click **Apply**.
9) Click **Save**.

See “Group attribute definition settings” on page 1400 for an explanation of group attribute definitions.

2. Map the federated repository property names to the LDAP attribute names.
a. Configure the LDAP repository to match the used LDAP attributes for a user.
1) Edit the file

```
{WAS_HOME}\profiles\{profileName}\config\cells\{cellName}\wim\config\wimconfig.xml
```

2) Look for the section in this file containing the LDAP repository configuration, For example,
a)

```
<config:repositories
xsi:type="config:LdapRepositoryType"
adapterClassName="com.ibm.ws.wim.adapter.ldap.LdapAda
pter" id="ldaprepo1" ...>
```

b)

```
<config:attributeConfiguration>
```

c)

```
...
```

d)

```
<config:attributes name="anLDAPattribute"
propertyName="aVMMattribute"/>
```

e)

```
...
<config:attributeConfiguration>
```

3) Add an element of type `config:attributes` to define the mapping between a given federated depository property name, such as `departmentNumber`, to a desired LDAP attribute name, such as `warehouseSection`.

Note: For all given federated depository properties, a one-to-one mapping is assumed. If no explicit mapping of the above type is defined, for example the federated repository property `departmentNumber`, the underlying LDAP attribute name, `departmentNumber` is assumed.

b. Configure the unsupported properties of the federated repository.

To indicate that a given federated repository property, such as `departmentNumber` is not supported by any LDAP attributes, you need to define the following type of element:

```
<config:repositories xsi:type="config:LdapRepositoryType"
adapterClassName="com.ibm.ws.wim.adapter.ldap.LdapAdapter"
id="ldaprepo1" ...>
<config:attributeConfiguration>
...
<config:propertiesNotSupported name=" departmentNumber"/>
...
<config:attributeConfiguration>
```

c. Configure the LDAP repository to match the used LDAP user membership attribute in the groups.
   1) In the administrative console, click **Security > Global security**.
   2) Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
   3) Under Related items, click **Manage repositories**.
   4) Select `ldaprepo1`
   5) Click **Group attribute defintions**.
   6) Click **Member attributes**.
   7) Check if your LDAP attributes (for example, `uniqueMember`) is specified for your LDAP objectclass (for example, `groupOfUniqueNames`).
      - If not specified, click **New** and add the pair (*objectclass / member attribute name*) that applies to your LDAP schema (for example, `uniqueMember / groupOfUniqueNames`
      - If specified, proceed.
   8) Click **Apply**.
   9) Click **Save**.

3. Map other LDAP settings by configuring a new base entry for the new LDAP repository.
   a. In the administrative console, click **Security > Global security**.
   b. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
   c. Click **Add Base Entry to Realm**.
   d. Select `ldaprepo1`.
   e. Specifiy:
      - The base entry within the federated repository realm, for example, `o=Default Organization`
      - The base entry within the LDAP repository, for example, `o=Default Organization`
   f. Click **Apply**.
   g. Click **Save**.

   For an explanation of base entries, see the Configuring supported entity types in a federated repository configuration topic.

Results

After completing these steps, your federated repository matches the LDAP server settings.

What to do next

Configuring Lightweight Directory Access Protocol in a federated repository configuration:

Follow this topic to configure Lightweight Directory Access Protocol (LDAP) settings in a federated repository configuration.

Before you begin

You have chosen among various ways to configure LDAP:

- "Configuring a single, Lightweight Directory Access Protocol repository in a new configuration under Federated repositories" on page 1331

- “Changing a federated repository configuration to include a single, Lightweight Directory Access Protocol repository only” on page 1333
- “Configuring multiple Lightweight Directory Access Protocol repositories in a federated repository configuration” on page 1334
- “Configuring a single built-in, file-based repository and one or more Lightweight Directory Access Protocol repositories in a federated repository configuration” on page 1335
- “Managing repositories in a federated repository configuration” on page 1380

About this task

At this point, you are viewing the LDAP repository configuration page of the administrative console.

Procedure

1. Enter a unique identifier for the repository in the Repository identifier field. This identifier uniquely identifies the repository within the cell, for example: LDAP1.
2. Select the type of LDAP server that is used from the Directory type list. The type of LDAP server determines the default filters that are used by WebSphere Application Server.

   IBM Tivoli Directory Server users can choose either IBM Tivoli Directory Server or SecureWay as the directory type. Use the IBM Tivoli Directory Server directory type for better performance. For a list of supported LDAP servers, see “Using specific directory servers as the LDAP server” on page 1275.
3. Enter the fully qualified host name of the primary LDAP server in the Primary host name field. You can enter either the IP address or the domain name system (DNS) name.
4. Enter the server port of the LDAP directory in the Port field. The host name and the port number represent the realm for this LDAP server in a mixed version nodes cell. If servers in different cells are communicating with each other using Lightweight Third Party Authentication (LTPA) tokens, these realms must match exactly in all the cells.

   The default value is 389, which is not a Secure Sockets Layer (SSL) connection. Use port 636 for a Secure Sockets Layer (SSL) connection. For some LDAP servers, you can specify a different port for a non-SSL or SSL connection. If you do not know the port to use, contact your LDAP server administrator.

   If multiple WebSphere Application Servers are installed and configured to run in the same single sign-on domain, or if WebSphere Application Server interoperates with a previous version of WebSphere Application Server, then it is important that the port number match all configurations. For example, if the LDAP port is explicitly specified as 389 in a Version 5.x or 6.0.x configuration, and WebSphere Application Server at Version 6.1 is going to interoperate with the Version 5.x or 6.0.x server, then verify that port 389 is specified explicitly for the Version 6.1 server.
5. Optional: Enter the host name of the failover LDAP server in the Failover host name field. You can specify a secondary directory server to be used in the event that your primary directory server becomes unavailable. After switching to a secondary directory server, LDAP repository attempts to reconnect to the primary directory server every 15 minutes.
6. Optional: Enter the port of the failover LDAP server in the Port field and click **Add**. The default value is 389, which is not a Secure Sockets Layer (SSL) connection. Use port 636 for a Secure Sockets Layer (SSL) connection. For some LDAP servers, you can specify a different port for a non-SSL or SSL connection. If you do not know the port to use, contact your LDAP server administrator.
7. Optional: Select the type of *referral*. A referral is an entity that is used to redirect a client request to another LDAP server. A referral contains the names and locations of other objects. It is sent by the server to indicate that the information that the client requested can be found at another location, possibly at another server or several servers. The default value is ignore.

   ignore
   Referrals are ignored.

   follow Referrals are followed automatically.

8. Optional: Enter the bind DN name in the Bind distinguished name field, for example, cn=root. The bind DN is required if anonymous binds are not possible on the LDAP server to obtain user and group information or for write operations. In most cases, bind DN and bind password are needed. However, when anonymous bind can satisfy all of the required functions, bind DN and bind password are not needed. If the LDAP server is set up to use anonymous binds, leave this field blank. If a name is not specified, the application server binds anonymously.

   Note: To create LDAP queries or to browse, an LDAP client must bind to the LDAP server using the distinguished name (DN) of an account that has the authority to search and read the values of LDAP attributes, such as user and group information. The LDAP administrator ensures that **read access privileges** are set for the bind DN. Read access privileges allow access to the subtree of the base DN and ensure that searches of user and group information are successful.

   The directory server provides an operational attribute in each directory entry (for example, the IBM Directory Server uses ibm-entryUuid as the operational attribute). The value of this attribute is a universally unique identifier (UUID), which is chosen automatically by the directory server when the entry is added, and is expected to be unique: no other entry with the same or different name would have this same value. Directory clients may use this attribute to distinguish objects identified by a distinguished name or to locate an object after renaming. **Ensure that the bind credentials have the authority to read this attribute.**
9. Optional: Enter the password that corresponds to the bind DN in the Bind password field.
10. Optional: Enter the property names to use to log into WebSphere Application Server in the Login properties field. This field takes multiple login properties, delimited by a semicolon (;). For example, `uid;mail`.

    All login properties are searched during login. If multiple entries or no entries are found, an exception is thrown. For example, if you specify the login properties as `uid;mail` and the login ID as Bob, the search filter searches for uid=Bob or mail=Bob. When the search returns a single entry, then authentication can proceed. Otherwise, an exception is thrown.

    Note: If you define multiple login properties, then the first login property is programmatically mapped to the federated repositories principalName property. For example, if you set `uid;mail` as the login properties, the LDAP attribute uid value is mapped to the federated repositories principalName property. If you define multiple login properties, after login, the first login property is returned as the value of the principalName property. For example, if you pass `joe@yourco.com` as the principalName value and the login properties are configured as uid;mail, the principalName is returned as joe.
11. Optional: Select the certificate map mode in the Certificate mapping field. You can use the X.590 certificates for user authentication when LDAP is selected as the repository. The Certificate mapping field is used to indicate whether to map the X.509 certificates into an LDAP directory user by EXACT_DN or CERTIFICATE_FILTER. If EXACT_DN is selected, the DN in the certificate must exactly match the user entry in the LDAP server, including case and spaces.
12. If you select **CERTIFICATE_FILTER** in the Certificate mapping field, specify the LDAP filter for mapping attributes in the client certificate to entries in LDAP.

    If more than one LDAP entry matches the filter specification at run time, authentication fails because the result is an ambiguous match. The syntax or structure of this filter is:

    ```
    LDAP attribute=${Client certificate attribute}
    ```

    For example, `uid=${SubjectCN}`.

    The left side of the filter specification is an LDAP attribute that depends on the schema that your LDAP server is configured to use. The right side of the filter specification is one of the public attributes in your client certificate. The right side must begin with a dollar sign ($) and open bracket ({) and end with a close bracket (}). You can use the following certificate attribute values on the right side of the filter specification. The case of the strings is important:
    - `${UniqueKey}`

- ${PublicKey}
- ${PublicKey}
- ${Issuer}
- ${NotAfter}
- ${NotBefore}
- ${SerialNumber}
- ${SigAlgName}
- ${SigAlgOID}
- ${SigAlgParams}
- ${SubjectCN}
- ${Version}

13. Optional: Select the **Require SSL communications** option if you want to use Secure Sockets Layer communications with the LDAP server.

    If you select the **Require SSL communications** option, you can select either the **Centrally managed** or **Use specific SSL alias** option.

    Centrally managed

    Enables you to specify an SSL configuration for a particular scope, such as the cell, node, server, or cluster in one location. To use the Centrally managed option, you must specify the SSL configuration for the particular set of endpoints. The Manage endpoint security configurations and trust zones panel displays all of the inbound and outbound endpoints that use the SSL protocol. If you expand the Inbound or Outbound section of the panel and click the name of a node, you can specify an SSL configuration that is used for every endpoint on that node. For an LDAP registry, you can override the inherited SSL configuration by specifying an SSL configuration for LDAP. To specify an SSL configuration for LDAP, complete the following steps:

    a. Click **Security > SSL certificate and key management > Manage endpoint security configurations and trust zones**.

    b. Expand **Outbound >** *cell_name* **> Nodes >** *node_name* **> Servers >** *server_name* **> LDAP**.

    Use specific SSL alias

    Select the **Use specific SSL alias** option if you intend to select one of the SSL configurations in the menu that follows the option.

    This configuration is used only when SSL is enabled for LDAP. The default is *DefaultSSLSettings*. To modify or create a new SSL configuration, complete the following steps:

    a. Click **Security > SSL certificate and key management**.

    b. Under Configuration settings, click **Manage endpoint security configurations and trust zones >** *configuration_name*.

    c. Under Related items, click **SSL configurations**.

14. Click **OK**.

Results

After completing these steps, your LDAP repository settings are configured.

What to do next

Return to the appropriate task to complete the steps for your federated repository configuration:
- “Configuring a single, Lightweight Directory Access Protocol repository in a new configuration under Federated repositories” on page 1331
- “Changing a federated repository configuration to include a single, Lightweight Directory Access Protocol repository only” on page 1333

- “Configuring multiple Lightweight Directory Access Protocol repositories in a federated repository configuration” on page 1334
- “Configuring a single built-in, file-based repository and one or more Lightweight Directory Access Protocol repositories in a federated repository configuration” on page 1335
- “Managing repositories in a federated repository configuration” on page 1380

Lightweight Directory Access Protocol repository configuration settings:

Use this page to configure secure access to a Lightweight Directory Access Protocol (LDAP) repository with optional failover servers.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Repository identifier:

Specifies a unique identifier for the LDAP repository. This identifier uniquely identifies the repository within the cell, for example: LDAP1.

Directory type:

Specifies the type of LDAP server to which you connect.

Expand the drop-down list to display a list of LDAP directory types.

Primary host name:

Specifies the host name of the primary LDAP server. This host name is either an IP address or a domain name service (DNS) name.

Port:

Specifies the LDAP server port.

The default value is 389, which is not a Secure Sockets Layer (SSL) connection. Use port 636 for a Secure Sockets Layer (SSL) connection. For some LDAP servers, you can specify a different port for a non-SSL or SSL connection. If you do not know the port to use, contact your LDAP server administrator.

Data type:	Integer	
Default:	389	
Range:	389, which is not a Secure Sockets Layer (SSL) connection	636, which is a Secure Sockets Layer (SSL) connection

Failover host name:

Specifies the host name of the failover LDAP server.

You can specify a secondary directory server to be used in the event that your primary directory server becomes unavailable. After switching to a secondary directory server, the LDAP repository attempts to reconnect to the primary directory server every 15 minutes.

Port:

Specifies the port of the failover LDAP server.

The default value is 389, which is not a Secure Sockets Layer (SSL) connection. Use port 636 for a Secure Sockets Layer (SSL) connection. For some LDAP servers, you can specify a different port for a non-SSL or SSL connection. If you do not know the port to use, contact your LDAP server administrator.

Data type:	Integer	
Range:	389, which is not a Secure Sockets Layer (SSL) connection	636, which is a Secure Sockets Layer (SSL) connection

Support referrals to other LDAP servers:

Specifies how referrals that are encountered by the LDAP server are handled.

A referral is an entity that is used to redirect a client request to another LDAP server. A referral contains the names and locations of other objects. It is sent by the server to indicate that the information that the client requested can be found at another location, possibly at another server or several servers. The default value is ignore.

Default:	ignore
Range:	**ignore** Referrals are ignored.
	follow Referrals are followed automatically.

Support for repository change tracking:

Specifies the type of support for repository change tracking, wih is one of the following:

none Specifies there is no change tracking support for this repository.

native Specifies that the repository's native change tracking mechanism is used by virtual member manager to return changed entities.

Custom properties:

Specifies arbitrary name and value pairs of data. The name is a property key and the value is a string value that can be used to set internal system configuration properties.

Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

Bind distinguished name:

Specifies the distinguished name (DN) for the application server to use when binding to the LDAP repository.

If no name is specified, the application server binds anonymously. In most cases, bind DN and bind password are needed. However, when anonymous bind can satisfy all of the required functions, bind DN and bind password are not needed.

Bind password:

Specifies the password for the application server to use when binding to the LDAP repository.

Login properties:

Specifies the property names to use to log into the application server.

This field takes multiple login properties, delimited by a semicolon (;). For example, `uid;mail`. All login properties are searched during login. If multiple entries or no entries are found, an exception is thrown. For example, if you specify the login properties as `uid;mail` and the login ID as Bob, the search filter searches for uid=Bob or mail=Bob. When the search returns a single entry, then authentication can proceed. Otherwise, an exception is thrown.

Note: If you define multiple login properties, then the first login property is programmatically mapped to the federated repositories principalName property. For example, if you set `uid;mail` as the login properties, the LDAP attribute uid value is mapped to the federated repositories principalName property. If you define multiple login properties, after login, the first login property is returned as the value of the principalName property. For example, if you pass `joe@yourco.com` as the principalName value and the login properties are configured as uid;mail, the principalName is returned as joe.

LDAP attribute for Kerberos principal name:

Specifies the LDAP attribute for Kerberos principal name. This field can be modified when Kerberos is configured and it is one of the active or preferred authentication mechanisms.

Certificate mapping:

Specifies whether to map X.509 certificates into an LDAP directory by EXACT_DN or CERTIFICATE_FILTER. Specify CERTIFICATE_FILTER to use the specified certificate filter for the mapping.

Certificate filter:

Specifies the filter certificate mapping property for the LDAP filter. The filter is used to map attributes in the client certificate to entries in the LDAP repository.

If more than one LDAP entry matches the filter specification at run time, authentication fails because the result is an ambiguous match. The syntax or structure of this filter is:

```
LDAP attribute=${Client certificate attribute}
```

An example of a simple certificate filter is: `uid=${SubjectCN}`.

You can also specify multiple properties and values as part of the certificate filter. Two examples of complex certificate filters are:

```
(&(cn=${IssuerCN}) (employeeNumber=${SerialNumber})

(& (issuer=${IssuerDN}) (serial=${SerialNumber}) (subjectdn=${SubjectDN}))
```

The left side of the filter specification is an LDAP attribute that depends on the schema that your LDAP server is configured to use. The right side of the filter specification is one of the public attributes in your client certificate. You can also use the **UniqueKey** certificate variable, which consists of the base64-encoding of the MD5 hash of the subject DN and issuer DN. The right side must begin with a

dollar sign ($) and open bracket ({) and end with a close bracket (}). You can use the following certificate attribute values on the right side of the filter specification. The case of the strings is important:
- `${UniqueKey}`
- `${PublicKey}`
- `${IssuerDN}`
- `${Issuer`*xx*`}` where *xx* is replaced by the characters that represent any valid component of the Issuer Distinguished Name. For example, you might use `${IssuerCN}` for the Issuer Common Name.
- `${NotAfter}`
- `${NotBefore}`
- `${SerialNumber}`
- `${SigAlgName}`
- `${SigAlgOID}`
- `${SigAlgParams}`
- `${SubjectDN}`
- `${Subject`*xx*`}` where *xx* is replaced by the characters that represent any valid component of the Subject Distinguished Name. For example, you might use `${SubjectCN}` for the Subject Common Name.
- `${Version}`

Require SSL communications:

Specifies whether secure socket communication is enabled to the LDAP server.

When enabled, the Secure Sockets Layer (SSL) settings for LDAP are used, if specified.

Centrally managed:

Specifies that the selection of an SSL configuration is based upon the outbound topology view for the Java Naming and Directory Interface (JNDI) platform.

Centrally managed configurations support one location to maintain SSL configurations, rather than spreading them across the configuration documents.

Default:	Enabled
Range:	Enabled or Disabled

Use specific SSL alias:

Specifies the SSL configuration alias to use for LDAP outbound SSL communications.

This option overrides the centrally managed configuration for the JNDI platform.

Migrating a stand-alone LDAP repository to a federated repositories LDAP repository configuration:

When configuring the security for your application server, you might need to migrate a stand-alone LDAP registry to a federated repositories LDAP repository configuration.

Before you begin

Note the specifications of your stand-alone LDAP repository that you want to migrate, for reference when configuring the LDAP repository in federated repositories. To access these fields, on the administrative console, click **Security > Global security**. To access these fields in a multiple security domain environment, click **Security > Security domains > DomainName**, and then, under Security Attributes, expand User Realm, and click **Customize for this domain**. Select a Realm type. and then click **Configure**.

The following table shows the administrative console panels and fields of the stand-alone LDAP repository configuration and their corresponding fields in a federated repositories LDAP repository configuration for mapping.

Table 86. Mapping between a stand-alone LDAP repository configuration and a federated repositories LDAP repository configuration. This table illustrates the mapping between a stand-alone LDAP repository configuration and a federated repositories LDAP repository configuration.

Stand-alone LDAP repository configuration	**LDAP repository in a federated repositories configuration**
Global security > Standalone LDAP registry General properties – Primary administrative user name	Global security > Federated repositories General properties – Primary administrative user name
Global security > Standalone LDAP registry LDAP server – Type of LDAP server	Global security > Federated repositories > Manage repositories > *repository_ID* LDAP server – Directory Type
Global security > Standalone LDAP registry LDAP server – Host	Global security > Federated repositories > Manage repositories > *repository_ID* LDAP server – Primary host name
Global security > Standalone LDAP registry LDAP server – Port	Global security > Federated repositories > Manage repositories > *repository_ID* LDAP server – Port
Global security > Standalone LDAP registry LDAP server – Failover hosts	Global security > Federated repositories > Manage repositories > *repository_ID* LDAP server – Failover server used when primary is not available
Global security > Standalone LDAP registry LDAP server – Base distinguished name (DN)	Global security > Federated repositories > Repository reference (Click Add Base entry to realm) General properties – Distinguished name of a base entry that uniquely identifies this set of entries in the realm and General properties – Distinguished name of a base entry in this repository
Global security > Standalone LDAP registry LDAP server – Search timeout	Global security > Federated repositories > Manage repositories > *repository_ID* > Performance General properties - Limit search time
Global security > Standalone LDAP registry LDAP server – Custom properties	Global security > Federated repositories > Custom properties
Global security > Standalone LDAP registry LDAP server – Server user identity	Global security > Federated repositories General properties – Server user identity
Global security > Standalone LDAP registry Security – Bind distinguished name (DN)	Global security > Federated repositories > Manage repositories > *repository_ID* Security – Bind distinguished name
Global security > Standalone LDAP registry Security – Bind password	Global security > Federated repositories > Manage repositories > *repository_ID* Security – Bind password

Table 86. Mapping between a stand-alone LDAP repository configuration and a federated repositories LDAP repository configuration (continued). This table illustrates the mapping between a stand-alone LDAP repository configuration and a federated repositories LDAP repository configuration.

Stand-alone LDAP repository configuration	LDAP repository in a federated repositories configuration
Global security > Standalone LDAP registry > Advanced Lightweight Directory Access Protocol (LDAP) user registry settings General properties – Kerberos user filter	Global security > Federated repositories > Manage repositories > *repository_ID* Security – LDAP attribute used for Kerberos principal name
Global security > Standalone LDAP registry > Advanced Lightweight Directory Access Protocol (LDAP) user registry settings General properties – Certificate map mode	Global security > Federated repositories > Manage repositories > *repository_ID* Security – Certificate mapping
Global security > Standalone LDAP registry > Advanced Lightweight Directory Access Protocol (LDAP) user registry settings General properties – Certificate filter	Global security > Federated repositories > Manage repositories > *repository_ID* Security – Certificate filter

The Realm name field under General Properties on the federated repositories LDAP configuration panel is not listed in the previous table because it does not have a one-to-one correspondence with a field in the stand-alone LDAP configuration panel. The host name and the port number represent the realm name for the standalone LDAP server in the WebSphere Application Server cell. For information on changing the realm name, see the topic Realm configuration settings.

The User Filter, Group Filter, User ID map, Group ID map, and Group member ID map fields also are not listed in the previous table as they do not have a one-to-one correspondence with fields in the federated repositories LDAP repository configuration panel. These LDAP attributes are set differently in the federated repositories LDAP repository configuration and involve multiple steps. These settings are explained in detail in the following sections and procedure.

About this task

Migrating from a stand-alone LDAP repository configuration to a federated repositories LDAP repository configuration involves migrating the configuration parameters, most of which are straight forward as shown in Table 1 in the previous section. Migrating the search filters is an important part of migrating a stand-alone LDAP repository configuration to a federated repository LDAP configuration; therefore, the concept and migration of LDAP search filters is described here in detail.

Stand-alone LDAP registry search filters follow the LDAP filter syntax, where you specify the attribute on which the search is based and its value.

The user filter is used for searching the registry for users. It is used to authenticate a user by using the attribute specified in the filter.

The group filter is used for searching the registry for groups. It specifies the property by which to look up groups.

Examples of commonly used LDAP user filters: In the following examples of search filters, %v is replaced with the corresponding search pattern of the user or group at run time.

```
(&(uid=%v)(objectclass=ePerson))
```

Searches for users where the uid attribute matches the specified search pattern of the ePerson object class.

```
(&(cn=%v)(objectclass=user))
```

Searches for users where the cn attribute matches the specified search pattern of the user object class.

```
(&(sAMAccountName=%v)(objectcategory=user))
```

Searches for users where the sAMAccountName attribute matches the specified search pattern of the user object category.

```
(&(userPrincipalName=%v)(objectcategory=user))
```

Searches for users where the userPrinciplalName attribute matches the specified search pattern of the user object category.

```
(&(mail=%v)(objectcategory=user))
```

Searches for users where the mail attribute matches the specified search pattern of the user object category.

```
(&(|(sAMAccountName=%v)(userPrincipalName=%v))(objectcategory=user))
```

Searches for users where the sAMAccountName or the userPrincipalName matches the specified search pattern of the user object category.

Examples of commonly used group filters:

```
(&cn=%v)(objectCategory=group)
```

Looks up groups based on their common names (cn).

```
(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))
```

Looks up groups based on their common names (cn) and by using the object class of either groupOfNames or groupOfUniqueNames.

As shown in these examples, a stand-alone LDAP registry search filter consists of LDAP attributes and object classes, based on which the search or login is performed.

You can also specify the LDAP attributes and object classes in the LDAP adapter configuration of federated repositories, but they are configured differently and provide more flexibility. In federated repositories the user is represented as PersonAccount entity type and group as Group entity type. Each entity type can have its own RDN (Relative Distinguished Name) property (rdnProperties) and object class. For example, the default RDN property of PersonAccount is uid, and the default RDN property of Group is cn. The default object class mapping depends on the LDAP server type. For example, for Tivoli Directory Server, the object class for PersonAccount is inetOrgPerson and object class for Group is groupOfNames. PersonAccount can also have login properties. When a user logs in or a search is performed for a user in a user registry, these login properties are matched with the pattern. For example, if the login properties are uid and mail, then for the search pattern, a*, all the users who match uid=a* or mail=a* are returned.

gotcha: You can specify the value of User ID Map property (userIdMap) of the stand-alone LDAP repository as the RDN property (rdnProperties) or the first login property (loginProperties) in federated repositories. Though you can set both the RDN property and the login property in federated repositories, it is sufficient if you set only the RDN property. The login property is

optional and you need to set it only if the login property is different from RDN property or if there are more than one login properties. If both the RDN property and login property are set, the login property takes precedence over RDN property.

Migrating search filters involves one or more of the following steps: setting the correct login properties, mapping the attributes of the back-end repository to the federated repositories properties, setting the object class, setting the search filter by using object class or object category, and setting the member or membership attribute. This mapping and configuration for federated repositories is maintained in the wimconfig.xml file.

The stand-alone LDAP registry search filter can be split into two parts:

- User or group attributes filter
- User or group object class or object category filter

For example, in the search filter, `(&(cn=%v)(objectclass=user))`:

- The attribute filter is (cn=%v)
- The object class filter is (objectclass=user)

These two filters are mapped separately in a federated repositories configuration:

- The attribute filter is mapped to the RDN properties or login properties configuration for user and to RDN properties configuration for group.
- The object class filter is mapped to the entity type configuration of the LDAP adapter.

The default attribute and object class mapping is set based on the LDAP server type but additional steps might be required to migrate these two filters:

- attribute filter:
  - Setting either or both the RDN property and login properties (if applicable)
  - Mapping the federated repository property to the LDAP attribute (if applicable)
- object class filter:
  - Setting the object class for entity type (if applicable)
  - Setting the search filter of entity type (if applicable)

Some of the steps in the following procedure include two examples. In these steps:

- Example 1 is applicable to the scenario where you are migrating the search filter `(&(cn=%v)(objectclass=ePerson))` from a stand-alone IBM Tivoli Directory Server LDAP repository to a federated repositories LDAP repository with the identifier LDAPTDS.
- Example 2 is applicable to the scenario where you are migrating the search filter `(&(|(sAMAccountName=%v)(userPrincipalName=%v))(objectcategory=user))` from a stand-alone Microsoft Active Directory LDAP repository to a federated repositories LDAP repository with the identifier LDAPAD. sAMAccountName and userPrincipalName attributes are not defined in federated repositories, so these attributes must be mapped to federated repository properties.

Procedure

1. Add the LDAP repository that you want to migrate to the federated repositories configuration.

   See Table 1 in the Before you begin section of this topic, and follow the steps described in the topic *Configuring a single, Lightweight Directory Access Protocol repository in a new configuration under Federated repositories*. These steps include links to other procedures that you must complete such as:

   - Adding an external repository in a federated repository configuration.
   - Configuring supported entity types in a federated repository configuration.
   - Configuring Lightweight Directory Access Protocol in a federated repository configuration.

After you complete these steps, the LDAP repository that you want to migrate will be successfully configured in the federated repository configuration.

2. Set the login properties (if applicable).

   Login properties are the property names that are used to log on to the WebSphere Application Server. You can specify multiple login properties by using the semicolon (;) as a delimiter. The federated repositories properties commonly used as login properties are uid, cn, sn, givenName, mail, and so on.

   To set login properties on the administrative console, follow the steps in the topic *Lightweight Directory Access Protocol repository configuration settings*, and apply the settings under the section, Login properties.

   Example 1: In the **Login properties** field, enter `cn`.

   Example 2: In the **Login properties** field, enter `uid;cn`.

   Complete Step 3 to map these properties to LDAP attributes.

3. Map the federated repository property to the LDAP attribute (if applicable).

   If the LDAP attribute is not a federated repository property, then the login property that you defined must be mapped to the LDAP attribute.

   a. In the administrative console, click **Global security > Federated repositories > Manage repositories >** *repository_ID*, and then, under Additional properties, click the **LDAP attributes** link.

   b. If the attribute mapping exists, you must first delete the existing mapping for the LDAP attribute, and then add a new mapping for the attribute. Select the checkbox next to the LDAP attribute name and click **Delete**.

   c. To add an attribute mapping, click **Add**, and select **Supported** from the drop-down menu. Enter the LDAP attribute name in the **Name** field, the federated repositories property name in the **Property name** field, and the entity type which applies the attribute mapping in the **Entity types** field.

   Example 1: Because the federated repository property cn is implicitly mapped to the cn LDAP attribute, no additional mapping is required.

   Example 2: Here the search filter includes two LDAP attributes, sAMAccountName and userPrincipalName.

   - For the LDAP server type, Active Directory, the LDAP attribute sAMAccountName is mapped by default to the federated repositories property, uid, as shown in the list of attributes on LDAP attributes panel. Therefore, you do not have to execute the addIdMgrLDAPAttr command to add an attribute configuration for sAMAccountName.
   - If an attribute mapping for the LDAP attribute userPrincipalName exists, then delete the existing attribute mapping before adding a new configuration.
     a. Select the checkbox next to **userPrincalName** and click **Delete**.
     b. Click **Add**, and select **Supported** from the drop-down menu.
     c. In the **Name** field, enter `userPrincipalName`.
     d. In the **Property name** field, enter `cn`.
     e. In the **Entity types** field, enter `PersonAccount`.

4. Set the object class for an entity type (if applicable).

   gotcha: Before executing this step, check the current mapping . If the object class mapping is already set, skip this step.

   To set the object class for an entity type on the administrative console, follow the steps in the topic *Lightweight Directory Access Protocol entity types settings*, and apply the following settings under the section, Object classes:

   - Specify `PersonAccount` as the entity type name for user filters
   - Specify `Group` as the entity type name for group filters.

Example 1: In the **Entity type** field, enter `PersonAccount`.

In the **Object classes** field, enter `ePerson`.

Example 2: In the **Entity type** field, enter `PersonAccount`.

In the **Object classes** field, enter `user`.

5. Set the search filter for the entity type (if applicable).

   Federated repositories performs the search based on the object class setting. To change this default setting and use object category as the filter, follow the steps in topic *Lightweight Directory Access Protocol entity types settings*, and apply the settings under the section, Search Filter.

   Example 1: Because the search is based on object class, no additional configuration is required.

   Example 2: In the **Search filter** field, enter `(objectcategory=user)`.

6. To migrate group filters, you must also configure the group attribute definition settings.

   The steps to configure the group attribute definition settings through the administrative console are specified in the topic *Locating user group memberships in a Lightweight Directory Access Protocol registry*, under the section, LDAP Registry within a Federated Repositories Registry. You can also use the wsadmin commands addIdMgrLDAPGroupDynamicMemberAttr or addIdMgrLDAPGroupMemberAttr that are described in the topic *IdMgrRepositoryConfig command group for the AdminTask object*.

7. Save your configuration changes
8. Restart the application server.

Results

After completing these steps, your LDAP repository is configured for use within the federated repositories configuration.

Adding an external repository in a federated repository configuration:

Follow this task to add an external repository into a federated repository configuration.

Procedure

1. If the Lightweight Directory Access Protocol (LDAP) repository that you want to add to your federated repository configuration is previously configured, select the corresponding Repository on the Repository reference panel. To access the Repository reference panel, complete the following steps:
   a. Click **Security > Global security**.
   b. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
   c. Click **Add base entry to realm**.
2. Enter a distinguished name for the realm base entry in the Distinguished name that uniquely identifies... field. This base entry must uniquely identify the external repository in the realm. If multiple repositories are included in the realm, use this field to define an additional distinguished name (DN) that uniquely identifies this set of entries within the realm. For example, repositories LDAP1 and LDAP2 might both use o=ibm,c=us as the base entry in the repository. Use the DN in this field to uniquely identify this set of entries in the realm. For example: o=ibm,c=us for LDAP1 and o=ibm2,c=us for LDAP2. The specified DN in this field maps to the LDAP DN of the base entry within the repository.
3. Enter the LDAP DN of the base entry within the repository in the Distinguished name of a base entry... field. The base entry indicates the starting point for searches in this LDAP directory server. This entry and its descendents are mapped to the subtree that is identified by this unique base name entry field. For example, for a user with a DN of `cn=John Doe, ou=Rochester, o=IBM, c=US`, specify the LDAP base entry as any of the following options:

`ou=Rochester, o=IBM, c=us` or `o=IBM, c=us` or `c=us`

In most cases, this LDAP DN is the same as the distinguished name for the realm base entry.

If this field is left blank, then the subtree defaults to the root of the LDAP repository. Consult your LDAP administrator to determine if your LDAP repository provides support to search from the root, or create users and groups under the root without defining a suffix beforehand.

In WebSphere Application Server, the distinguished name is normalized according to the LDAP specification. Normalization consists of removing spaces in the base distinguished name before or after commas and equal symbols. An example of a non-normalized base distinguished name is o = ibm, c = us or o=ibm, c=us. An example of a normalized base distinguished name is o=ibm,c=us.

4. If the LDAP repository that you want to add to your realm is not previously configured, complete the following steps:
   a. Click **Add Repository** on the Repository reference panel to configure the LDAP repository. See step 1 to access the Repository reference panel.
   b. Configure LDAP on the LDAP configuration panel, as described in “Configuring Lightweight Directory Access Protocol in a federated repository configuration” on page 1339.
   c. Select the new Repository on the Repository reference panel.
5. Click **OK**.

Results

You have added a new or previously configured external repository into your federated repository configuration.

What to do next

1. Before you can manage this account with Users and Groups, configure supported entity types as described in “Configuring supported entity types in a federated repository configuration” on page 1377.
2. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
3. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
4. Save, stop, and restart all the product servers (deployment managers, nodes and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Configuring a property extension repository in a federated repository configuration:

Follow this task to configure a property extension repository to store attributes that cannot be stored in your Lightweight Directory Access Protocol (LDAP) server.

About this task

For security and business reasons, you might want to prohibit write operations to your repositories. However, applications calling the federated repository configuration might need to store additional properties for the entities. A federated repository configuration provides a *property extension repository*, which is a database regardless of the type of main profile repositories, for a propertylevel join configuration. For example, a company that uses an LDAP directory for its internal employees and a database for external customers and business partners might not allow write access to its LDAP and its database. The company can use the *property extension repository* in a federated repository configuration to store additional properties for the people in those repositories, excluding the user ID. When an

application uses the federated repository configuration to retrieve an entry for a person, the federated repository configuration transparently joins the properties of the person that is retrieved from either the LDAP or the customer's database with the properties of the person that is retrieved from the property extension repository into a single logical person entry.

When you configure a property extension repository, you can supply a valid data source, a direct connection configuration, or both. The system first tries to connect by way of the data source. If the data source is not available, then the system uses the direct access configuration.

Restriction: You cannot configure a property extension repository in a mixed version deployment manager cell.

Procedure

1. Configure the WebSphere Application Server data source. See “Configuring the WebSphere Application Server data source” on page 1373.
2. If you are adding new properties (including properties that are stored in the property extension repository) to the schema, you must do the following before you create the property extension repository.
   a. Open or create the `wimxmlextension.xml` file under the *profile_root*`\config\cells\`*cell_name*`\wim\model` directory.

      Attention: Make sure the editor is on the deployment manager node.

   b. Add the schema definition of the new property. The following sample `wimxmlextension.xml` file adds a new property called ibmotherEmail to both the Person and PersonAccount entity types. This new property type is "String" and it is multiplevalued.

```
<sdo:datagraph xmlns:sdo="commonj.sdo"
    xmlns:wim="http://www.ibm.com/websphere/wim">
 <wim:schema>
  <wim:propertySchema
   nsURI="http://www.ibm.com/websphere/wim"
   dataType="String"
   multiValued="true"
   propertyName="ibm-otherEmail">
     <wim:applicableEntityTypeNames>Person
    </wim:applicableEntityTypeNames>
    <wim:applicableEntityTypeNames>PersonAccount
    </wim:applicableEntityTypeNames>
  </wim:propertySchema>
  <wim:propertySchema
   nsURI="http://www.ibm.com/websphere/wim"
   dataType="String"
   multiValued="true"
   propertyName="ibm-personalTitle">
    <wim:applicableEntityTypeNames>Person
    </wim:applicableEntityTypeNames>
    <wim:applicableEntityTypeNames>PersonAccount
    </wim:applicableEntityTypeNames>
  </wim:propertySchema>
   <wim:propertySchema
   nsURI="http://www.ibm.com/websphere/wim"
   dataType="String"
   multiValued="true"
   propertyName="ibm-middleName">
    <wim:applicableEntityTypeNames>Person
    </wim:applicableEntityTypeNames>
    <wim:applicableEntityTypeNames>PersonAccount
    </wim:applicableEntityTypeNames>
  </wim:propertySchema>
   <wim:propertySchema
   nsURI="http://www.ibm.com/websphere/wim"
   dataType="String" multiValued="true"
   propertyName="ibm-generationQualifier">
    <wim:applicableEntityTypeNames>Person
    </wim:applicableEntityTypeNames>
    <wim:applicableEntityTypeNames>PersonAccount
    </wim:applicableEntityTypeNames>
    </wim:propertySchema>
  <wim:propertySchema
```

```
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="false"
 propertyName="ibm-regionalLocale">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
</wim:propertySchema>
 <wim:propertySchema
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="false"
 propertyName="ibm-timeZone">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
</wim:propertySchema>
 <wim:propertySchema
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="false"
 propertyName="ibm-preferredCalendar">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
  </wim:propertySchema>
<wim:propertySchema
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="false"
 propertyName="ibm-alternativeCalendar">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
</wim:propertySchema>
 <wim:propertySchema
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="false"
 propertyName="ibm-firstDayOfWeek">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
  </wim:propertySchema>
 <wim:propertySchema
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="false"
 propertyName="ibm-firstWorkDayOfWeek">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
  </wim:propertySchema>
 <wim:propertySchema
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="false"
 propertyName="ibm-gender">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
  </wim:applicableEntityTypeNames>
  </wim:propertySchema>
<wim:propertySchema
 nsURI="http://www.ibm.com/websphere/wim"
 dataType="String"
 multiValued="true"
 propertyName="ibm-hobby">
  <wim:applicableEntityTypeNames>Person
  </wim:applicableEntityTypeNames>
  <wim:applicableEntityTypeNames>PersonAccount
```

```
    </wim:applicableEntityTypeNames>
  </wim:propertySchema>
 </wim:schema>
</sdo:datagraph>
```

Available data types are defined in `com.ibm.websphere.wim.SchemaConstants`. For example:

```
/**
  * Instance Class: java.lang.String
  */
 String DATA_TYPE_STRING = "String";
 /**
  * Instance Class: int
  */
 String DATA_TYPE_INT = "Int";
 /**
  * Instance Class: java.lang.Object
  */
 String DATA_TYPE_DATE = "Date";
 /**
  * Instance Class: dobjava.lang.Object
  */
 String DATA_TYPE_ANY_SIMPLE_TYPE = "AnySimpleType";
 /**
  * Instance Class: java.lang.String
  */
 String DATA_TYPE_ANY_URI = "AnyURI";
 /**
  * Instance Class: java.lang.boolean
  */
 String DATA_TYPE_BOOLEAN = "Boolean";
 /**
  * Instance Class: long
  */
 String DATA_TYPE_LONG = "Long";
 /**
  * Instance Class: double
  */
 String DATA_TYPE_DOUBLE = "Double";
 /**
  * Instance Class: short
  */
 String DATA_TYPE_SHORT = "Short";
```

c. Add the new property to the property extension repository. Before running the `setupIdMgrPropertyExtensionRepositoryTables` command, add the new properties into *install_root*`/etc/wim/setup/wimlaproperties.xml`.

d. Follow the example inside this file to define the new property definitions. The schema file for `wimlaproperties.xml` is `wimdbproperty.xsd` and is in the same directory. It can be used for reference.

3. Run the `setupIdMgrPropertyExtensionRepositoryTables` command to create the property extension repository and to add the new properties.
4. Set up the property extension repository using wsadmin by following the procedure discussed in “Setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands” on page 1359; ignore the "Before you begin" options.
5. Configure the property extension repository by completing the following steps:
   a. In the administrative console, click **Security > Global security**.
   b. Under User account repository, select **Federated repositories**, and click **Configure**.
   c. Click **Property extension repository**.
   d. Supply the name of the data source in the Data source name field.
   e. Select the type of database that is used for the property extension repository.
   f. Supply the name of the Java database connectivity (JDBC) driver in the JDBC driver field.

      Values include:

      DB2 `com.ibm.db2.jcc.DB2Driver`

      Oracle
      `oracle.jdbc.driver.OracleDriver`

      Informix
      `com.informix.jdbc.IfxDriver`

Microsoft SQL Server
com.microsoft.jdbc.sqlserver.SQLServerDriver

Derby org.apache.derby.jdbc.EmbeddedDriver

g. Supply the database URL that is used to access the property extension repository with JDBC in the Database URL field. Use an alphanumeric text string that conforms to the standard JDBC URL syntax.

Values include:

DB2 jdbc:db2://<*hostname*>:<*port*>/<*DB2location*>

Oracle
jdbc:oracle:thin:@<*hostname*>:<*port*>:<*dbname*>

Derby jdbc:derby:c:\derby\wim

Microsoft SQL Server
jdbc:microsoft:sqlserver://<*hostname*>:1433;databaseName=wim;selectmethod=cursor;

Informix
jdbc:informixsqli://<*hostname*>:1526/wim:INFORMIXSERVER=<*IFXServerName*>;

h. Supply the user name of the database administrator in the Database administrator user name field.

i. Supply the password of the database administrator in the Password field.

j. Specify the entity retrieval limit in the Entity retrieval limit field. The entity retrieval limit is the maximum number of entities that the system can retrieve from the property extension repository with a single database query. The default value is 200.

k. Click **OK**.

Results

After completing these steps, your federated repository configuration, which includes a property extension repository, is configured.

What to do next

1. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** on the Global security panel.
2. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Property extension repository settings:

Use this page to configure a property extension repository that is used to store attributes that cannot be stored in existing repositories.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Additional properties, click **Property extension repository**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Data source name:

Specifies the Java Naming and Directory Interface (JNDI) name of the data source that is used to access the property extension repository.

Default: `jdbc/wimDS`

Database type:

Specifies the type of database that is used for the property extension repository.

Default: `DB2`

JDBC driver:

Specifies the Java Database Connectivity (JDBC) driver that is used to access the entry mapping repository.

Values include:

DB2 `COM.ibm.db2.jcc.DB2Driver`

Oracle
`oracle.jdbc.driver.OracleDriver`

Informix
`com.informix.jdbc.IfxDriver`

DataDirect Connect
`com.ddtek.jdbc.sqlserver.SQLServerDriver`

Derby `org.apache.derby.jdbc.EmbeddedDriver`

Microsoft SQL Server
`com.microsoft.sqlserver.jdbc.SQLServerDriver`

Database URL:

Specifies the web address for the property extension repository.

Values include:

DB2 `jdbc:db2:wim`

Informix
`jdbc:informix-sqli://`*`host_name`*`:`*`port`*`/wim:INFORMIXSERVER=`*`IFXServerName`*`;`

DataDirect Connect
jdbc:datadirect:sqlserver://*host_name*:1433;databaseName=wim;selectMethod=cursor;

Derby `jdbc:derby:c:\derby\wim`

Oracle
`jdbc:oracle:thin:@`*`host_name`*`:`*`port`*`:`*`dbname`*

Microsoft SQL Server
`jdbc:sqlserver://`*`host_name`*`:1433;databaseName=wim;selectMethod=cursor;`

Database administrator user name:

Specifies the user name of the database administrator that is used to access the property extension repository.

Password:

Specifies the password that is used to enable the database administrator to access the property extension repository.

Entity retrieval limit:

Specifies the maximum number of entities that the system can retrieve from the property extension repository with a single database query.

Data type:	Integer
Default:	200

Setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands:

You can set up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands.

Before you begin

If you are setting up an entry mapping repository, begin with the steps described in “Configuring an entry mapping repository in a federated repository configuration” on page 1374.

If you are setting up a property extension repository, begin with the steps described in “Configuring a property extension repository in a federated repository configuration” on page 1353.

About this task

When you create a repository, use the appropriate wsadmin commands to define the database schema and to populate the database property definitions.

Procedure

1. Create the database. You can use any relational database product. The following examples give you tips for specific vendors.
   a. For **DB2**, open a DB2 command window or command center and enter the following:

```
db2 create database <name> using codeset UTF-8 territory US
```

   Enter the following database tuning commands:

```
db2 update database configuration for <name> using applheapsz 1024
db2 update database configuration for <name> using stmtheap 4096
db2 update database configuration for <name> using app_ctl_heap_sz 2048
db2 update database configuration for <name> using locklist 1024
db2 update database configuration for <name> using indexrec RESTART
db2 update database configuration for <name> using logfilsiz 1000
db2 update database configuration for <name> using logprimary 12
db2 update database configuration for <name> using logsecond 10
db2 update database configuration for <name> using sortheap 2048
db2set DB2_RR_TO_RS=yes
```

   b. Optional: For **Informix** databases using `dbaccess`, enter the following command:

```
CREATE DATABASE <name> WITH BUFFERED LOG
```

   c. Optional: For **Oracle** databases, the database should already exist during Oracle installation (for example, `orcl`).
2. Run the `setupIdMgrEntryMappingRepositoryTables` command, the `setupIdMgrPropertyExtensionRepositoryTables` command, or the `setupIdMgrDBTables` command (for custom registry repositories) by doing the following:
   a. Start WebSphere Application Server.
   b. Open a command window and go to the `<WAS>/Profiles/<PROFILE_NAME>bin` directory.

c. Start wsadmin.
d. Type the necessary commands as described below.

What to do next

Using these commands, you can:
- Specify the arguments on the command line.
- Specify the arguments in a file.

The `-file` option enables you to specify a file in which some or all of the parameters are specified. To use the `-file` argument on the command line, enter the full path to the file. Parameters in the file must be specified in `key=value` pairs and each must be on its own line. If a parameter is specified on both the command line and in the file, the value on the command line takes precedence.

Tips for diagnosing argument errors:
- If an argument is not properly specified on the command line or in the file, a message is returned which states that the argument was not properly specified. This might mean that the argument was not specified at all or was required for a given configuration but was not specified.
- If the argument was not specified at all, check that the parameter is specified on the command line or in the file, and that it is properly spelled and has matching case.
- If the argument was required for a given configuration but was not specified, it is possible that a value is not required solely by the command but is required for the type of database and configuration you are setting.

  For example, if you set the `dn`, `wasAdminId`, or `wasAdminPassword` parameters, you must also specify the `dbDriver` parameter.

  Additionally, if the `dn`, `wasAdminId` or `wasAdminPassword` parameters are specified, and the `databaseType` is not a Apache Derby v10.2 database, then the `dbAdminId` and `dbAdminPassword` parameters must also be specified.

The setupIdMgrDBTables command:

The setupIdMgrDBTables command creates, and populates the tables in the database that you previously created. Arguments are case-sensitive, both through the command line and the file.

Parameters:

schemaLocation (String, Required)
The location of the `<WAS>/etc/wim/setup` directory.

dbPropXML (String)
The location of database repository property definition XML file.

databaseType (String, Required)
The type of database. Supported databases are db2, oracle, informix, derby, sqlserver, db2zos, and db2iseries.

dbURL (String, Required)
The database URL for direct access mode. For example: jdbc:db2:wim.

dbDriver (String)
The name of the database driver. For example: com.ibm.db2.jcc.DB2Driver.

dbAdminId (String)
The database administrator ID for direct access mode. For example: db2admin.

Note: For a Apache Derby v10.2 embedded database, dbAdminId is not required.

dbAdminPassword (String)
The password associated with the dbAdminId.

Note: For a Apache Derby v10.2 embedded database, dbAdminPassword is not required.

dn (String)
The default organization uniqueName to replace. For example: `o=yourco`. If it is not set, `o=Default Organization` is used.

wasAdminId (String)
The WebSphere Application Server admin user ID. The ID should be a short name, not a uniqueName. For example: wasadmin. After creation, the uniqueName is `uid=wasadmin, <defaultOrg>`.

wasAdminPassword (String)
The WebSphere Application Server admin user password. If `wasAdminId` is set, then this parameter is mandatory.

saltLength (Integer)
The salt length of the randomly generated salt for password hashing.

encryptionKey (String)
The password encryption key. Set the password encryption key to match the encryption key in the wimconfig.xml file for the repository. If the encryption key is not set, the default is used.

derbySystemHome (String)
The home location of the Apache Derby v10.2 system if you are setting up a Apache Derby v10.2 database.

reportSqlError (String)
Specifies whether to report SQL errors while setting up databases.

file (String)
The full path to a file containing the input parameters. Each input parameter must match a corresponding parameter as it would be typed on the command line, and it must be placed in a `key=value` pair. Each pair must be on a separate line.

dbSchema (String)
The database schema where you want to create the federated repository tables. The schema should exist in the database. The default value is the default schema of the database according to the database type. Typically, the default schema is the namespace of the current database user.

The deleteIdMgrDBTables command:

The deleteIdMgrDBTables command deletes the tables in the database.

Parameters:

schemaLocation (String, Required)
The location of the `<WAS>/etc/wim/setup` directory.

databaseType (String, Required)
The type of database. Supported databases are db2, oracle, informix, derby, sqlserver, db2zos, and db2iseries.

dbURL (String, Required)
The database URL for direct access mode. For example: jdbc:db2:wim.

dbDriver (String)
The name of the database driver. For example: com.ibm.db2.jcc.DB2Driver.

dbAdminId (String)
The database administrator ID for direct access mode. For example: db2admin.

Note: For a Apache Derby v10.2 embedded database, dbAdminId is not required.

dbAdminPassword (String)
The password associated with the dbAdminId.

Note: For a Apache Derby v10.2 embedded database, dbAdminPassword is not required.

derbySystemHome (String)
The home location of the Apache Derby v10.2 system if you are setting up a Apache Derby v10.2 database.

reportSqlError (String)
Specifies whether to report SQL errors while setting up databases.

file (String)
The full path to a file containing the input parameters. Each input parameter must match a corresponding parameter as it would be typed on the command line, and it must be placed in a `key=value` pair. Each pair must be on a separate line.

dbSchema (String)
The database schema from which you want to delete the federated repository tables. The schema should exist in the database. The default value is the default schema of the database according to the database type. Typically, the default schema is the namespace of the current database user.

The setupIdMgrPropertyExtensionRepositoryTables command:

The setupIdMgrPropertyExtensionRepositoryTables command sets up the property extension repository. The default behavior includes creating and populating the tables in the database.

This command is available in connected or local mode.

Parameters:

schemaLocation (String, Required)
The location of the *app_server_root*`/etc/wim/setup` directory.

laPropXML (String)
The location of the property extension repository definition XML file.

databaseType (String, Required)
The type of database. Supported databases are db2, oracle, informix, derby, sqlserver, db2zos, and db2iseries.

dbURL (String, Required)
The database URL for direct access mode. For example: jdbc:db2:wim.

dbAdminId (String)
The database administrator ID for direct access mode. For example: db2admin.

Note: For a Apache Derby v10.2 embedded database, dbAdminId is not required.

dbAdminPassword (String)
The password associated with the dbAdminId.

Note: For a Apache Derby v10.2 embedded database, dbAdminPassword is not required.

derbySystemHome (String)
The home location of the Apache Derby v10.2 system if you are setting up a Apache Derby v10.2 database.

reportSqlError (String)
Specifies whether to report SQL errors while setting up databases.

skipDBCreation (Boolean)
Specifies whether to create the tables in the property extension repository.

If you set this parameter value to false or do not specify a value, then the command follows the default behavior of creating and populating the tables in the database.

If you set this parameter value to true, manually set up the property extension repository before running this command so that the tables get populated. For more information on this manual process, see the appropriate topic on manually setting up the property extension repository for your database.

file (String)
The full path to a file containing the input parameters. Each input parameter must match a corresponding parameter as it would be typed on the command line, and it must be placed in a `key=value` pair. Each pair must be on a separate line.

dbSchema (String)
The database schema where you want to create the federated repository tables. The schema should exist in the database. The default value is the default schema of the database according to the database type. Typically, the default schema is the namespace of the current database user.

The deleteIdMgrPropertyExtensionRepositoryTables command:

The deleteIdMgrPropertyExtensionRepositoryTables command deletes the tables in the property extension database.

This command is available in the connected or local mode.

Parameters:

schemaLocation (String, Required)
The location of the `<WAS>/etc/wim/setup` directory.

databaseType (String, Required)
The type of database. Supported databases are db2, oracle, informix, derby, sqlserver, db2zos, and db2iseries.

dbURL (String, Required)
The database URL for direct access mode. For example: jdbc:db2:wim.

dbDriver (String)
The name of the database driver. For example: com.ibm.db2.jcc.DB2Driver.

dbAdminId (String)
The database administrator ID for direct access mode. For example: db2admin.

Note: For a Apache Derby v10.2 embedded database, dbAdminId is not required.

dbAdminPassword (String)
The password associated with the dbAdminId.

Note: For a Apache Derby v10.2 embedded database, dbAdminPassword is not required.

derbySystemHome (String)
The home location of the Apache Derby v10.2 system if you are setting up a Apache Derby v10.2 database.

reportSqlError (String)
Specifies whether to report SQL errors while setting up databases.

file (String)
The full path to a file containing the input parameters. Each input parameter must match a

corresponding parameter as it would be typed on the command line, and it must be placed in a `key=value` pair. Each pair must be on a separate line.

dbSchema (String)
The database schema from which you want to delete the federated repository tables. The schema should exist in the database. The default value is the default schema of the database according to the database type. Typically, the default schema is the namespace of the current database user.

The setupIdMgrEntryMappingRepositoryTables command:

The setupIdMgrEntryMappingRepositoryTables command sets up the entry mapping repository, which includes creating and populating the tables of the repository.

Parameters:

schemaLocation (String, Required)
The location of the `<WAS>/etc/wim/setup` directory.

databaseType (String, Required)
The type of database. Supported databases are db2, oracle, informix, derby, sqlserver, db2zos, and db2iseries.

dbURL (String, Required)
The database URL for direct access mode. For example: jdbc:db2:wim.

dbDriver (String)
The name of the database driver. For example: com.ibm.db2.jcc.DB2Driver.

dbAdminId (String)
The database administrator ID for direct access mode. For example: db2admin.

Note: For a Apache Derby v10.2 embedded database, dbAdminId is not required.

dbAdminPassword (String)
The password associated with the dbAdminId.

Note: For a Apache Derby v10.2 embedded database, dbAdminPassword is not required.

derbySystemHome (String)
The home location of the Apache Derby v10.2 system if you are setting up a Apache Derby v10.2 database.

reportSqlError (String)
Specifies whether to report SQL errors while setting up databases.

file (String)
The full path to a file containing the input parameters. Each input parameter must match a corresponding parameter as it would be typed on the command line, and it must be placed in a `key=value` pair. Each pair must be on a separate line.

dbSchema (String)
The database schema where you want to create the federated repository tables. The schema should exist in the database. The default value is the default schema of the database according to the database type. Typically, the default schema is the namespace of the current database user.

The deleteIdMgrEntryMappingRepositoryTables command:

The deleteIdMgrEntryMappingRepositoryTables command deletes the tables in the entry mapping repository.

Parameters:

schemaLocation (String, Required)
The location of the <WAS>/etc/wim/setup directory.

databaseType (String, Required)
The type of database. Supported databases are db2, oracle, informix, derby, sqlserver, db2zos, and db2iseries.

dbURL (String, Required)
The database URL for direct access mode. For example: jdbc:db2:wim.

dbDriver (String)
The name of the database driver. For example: com.ibm.db2.jcc.DB2Driver.

dbAdminId (String)
The database administrator ID for direct access mode. For example: db2admin.

Note: For a Apache Derby v10.2 embedded database, dbAdminId is not required.

dbAdminPassword (String)
The password associated with the dbAdminId.

Note: For a Apache Derby v10.2 embedded database, dbAdminPassword is not required.

derbySystemHome (String)
The home location of the Apache Derby v10.2 system if you are setting up a Apache Derby v10.2 database.

reportSqlError (String)
Specifies whether to report SQL errors while setting up databases.

file (String)
The full path to a file containing the input parameters. Each input parameter must match a corresponding parameter as it would be typed on the command line, and it must be placed in a key=value pair. Each pair must be on a separate line.

dbSchema (String)
The database schema from which you want to delete the federated repository tables. The schema should exist in the database. The default value is the default schema of the database according to the database type. Typically, the default schema is the namespace of the current database user.

Sample command line usage:

To set up a database using the command line, enter the following:

```
$AdminTask setupIdMgrDBTables {-schemaLocation "C:/WAS/etc/wim/setup" -dbPropXML
"C:/WAS/etc/wim/setup/wimdbproperties.xml" -databaseType db2
-dbURL jdbc:db2:wim -dbAdminId db2admin
-dbDriver com.ibm.db2.jcc.DB2Driver -dbAdminPassword db2adminPwd
-reportSqlError true}
```

To delete database tables using the command line, enter the following:

```
$AdminTask deleteIdMgrDBTables {-schemaLocation "C:/WAS/etc/wim/setup"
-databaseType db2  -dbURL jdbc:db2:wim -dbAdminId db2admin
-dbDriver com.ibm.db2.jcc.DB2Driver -dbAdminPassword db2adminPwd
-reportSqlError true}
```

To set up a property extension repository using the command line, enter the following:

```
$AdminTask setupIdMgrPropertyExtensionRepositoryTables {-schemaLocation
"C:/WAS/etc/wim/setup"
-laPropXML "C:/WAS/etc/wim/setup/wimlaproperties.xml" -databaseType db2
-dbURL jdbc:db2:wim -dbAdminId db2admin -dbDriver com.ibm.db2.jcc.DB2Driver
-dbAdminPassword db2adminPwd -reportSqlError true}
```

To delete a property extension repository using the command line, enter the following:

```
$AdminTask deleteIdMgrPropertyExtensionRepositoryTables {-schemaLocation "C:/WAS/etc/wim/setup "
-databaseType db2 -dbURL jdbc:db2:wim -dbAdminId db2admin -dbDriver
com.ibm.db2.jcc.DB2Driver -dbAdminPassword db2adminPwd -reportSqlError true}
```

To set up an entry mapping repository using the command line, enter the following:

```
$AdminTask setupIdMgrEntryMappingRepositoryTables {-schemaLocation "C:/WAS/etc/wim/setup"
-databaseType db2 -dbURL jdbc:db2:wim -dbAdminId db2admin -dbDriver
com.ibm.db2.jcc.DB2Driver -dbAdminPassword db2adminPwd -reportSqlError true}
```

To delete an entry mapping repository using the command line, enter the following:

```
$AdminTask deleteIdMgrEntryMappingRepositoryTables {-schemaLocation "C:/WAS/etc/wim/setup"
-databaseType db2 -dbURL jdbc:db2:wim -dbAdminId db2admin -dbDriver
com.ibm.db2.jcc.DB2Driver -dbAdminPassword db2adminPwd -reportSqlError true}
```

Sample CLI Usage using -file option:

To set up a database with the `-file` option using the example `params.txt` file below, enter the following:

```
$AdminTask setupIdMgrDBTables {–file C:/params.txt -dbPropXML
"C:/OverrideDBPropParam/wimdbproperties.xml"}
```

Params.txt

```
schemaLocation=C:/WAS/etc/wim/setup
dbPropXML=C:/Program Files/IBM/WebSphere/AppServer/profiles/default
/config/cells/mycell/wim/config/wimdbproperties.xml
laPropXML=C:/Program Files/IBM/WebSphere/AppServer/profiles/default
/config/cells/mycell/wim/config/wimlaproperties.xml
databaseType=db2
dbURL=jdbc:db2:wim
dbDriver=com.ibm.db2.jcc.DB2Driver
reportSqlError=true
dn=o=db.com
dbAdminId=db2admin
dbAdminPassword=dbPassword
wasAdminId=wasadmin
wasAdminPassword=wasadmin1
```

To set up a database with the `-file` option using a file only, enter the following:

```
$AdminTask setupIdMgrDBTables {-file C:/params.txt}
```

Note: The use of a file only works if `-file` is the only parameter specified on the command line. If other parameters are specified then the file is completely ignored, and only the parameters on the command line are used to execute the command.

Manually setting up the property extension repository for federated repositories:

You can use the createIdMgrPropExtDbTables script to create tables in the property extension repository for federated repositories.

Before you begin

The following databases are supported by the script when the database exists on a distributed operating system:

- IBM DB2
- Apache Derby
- IBM Informix Dynamic Server
- Oracle 11g
- Microsoft SQL Server

For a list of the supported database versions, see the IBM WebSphere Application Server detailed system requirements.

To use the IBM DB2 on z/OS or IBM DB2 on iSeries database, read about manually setting up the property extension repository in DB2.

If you do not have WebSphere Application Server installed on the same system on which you are setting up the database, you must copy the following files from a system where WebSphere Application Server is installed to the system on which you are setting up the database. Ensure that you replicate the same directory structure within the setup directory. The *db_type* variable represents one of the following directory names: `db2`, `oracle`, `informix`, `derby`, or `sqlserver`.

app_server_root\etc\wim\setup\bin\createIdMgrPropExtDbTables.sh
app_server_root\etc\wim\setup\bin\createIdMgrPropExtDbTables.bat
app_server_root\etc\wim\setup\lookaside*db_type*\dbclean.sql
app_server_root\etc\wim\setup\lookaside*db_type*\schema.sql
app_server_root\etc\wim\setup\lookaside*db_type*\primarykeys.sql
app_server_root\etc\wim\setup\lookaside*db_type*\indexes.sql
app_server_root\etc\wim\setup\lookaside*db_type*\references.sql
app_server_root\etc\wim\setup\lookaside\keys.sql
app_server_root\etc\wim\setup\lookaside\bootstrap.sql

Specifying the database schema:

You can specify the database schema where you want to create the federated repository tables when you are manually setting up the property extension repository.

If you want to use the default schema of the database, you must execute the following commands without specifying the DBSCHEMA parameter. Typically, the default schema is the namespace of the current database user.

Complete these steps to replace the schema variable in the SQL files with the actual database schema name. If WebSphere Application Server and the database are not on the same system, set the SCHEMA_LOCATION value to the location where you copied the SQL files.

Windows operating systems:

1. Open a command window.
2. Change to the *app_server_root*\etc\wim\setup directory.
3. Enter the following commands:

```
set SCHEMA_LOCATION=app_server_root\etc\wim\setup\lookaside
set DBTYPE=<db_type>
set DBSCHEMA=dbschemaname
set SCHEMA_DEST_LOCATION=<location where the updated SQL files with replaced variables should be copied>
ws_ant.bat -f app_server_root\etc\wim\setup\filterbuild.xml
```

   where the value of <*db_type*> is db2, derby, informix, oracle, or sqlserver.

Note: : If SCHEMA_DEST_LOCATION is not set, the updated SQL files are copied to a directory with the name as the value not substituted under the current directory. The output shows where the files are copied.

AIX, HP-UX, Linux, and Solaris operating systems:

1. Open a command window
2. Change to the *app_server_root*/etc/wim/setup directory.
3. Enter the following commands:

```
export SCHEMA_LOCATION=app_server_root/etc/wim/setup/lookaside
export DBTYPE=<db_type>
export DBSCHEMA=dbschemaname
export SCHEMA_DEST_LOCATION=<location where the updated SQL files with replaced variables should be copied>
ws_ant.sh -f app_server_root/etc/wim/setup/filterbuild.xml
```

   where the value of <*db_type*> is db2, derby, informix, oracle, or sqlserver.

Note: If SCHEMA_DEST_LOCATION is not set, the updated SQL files are copied to a directory with the name as the value not substituted under the current directory. The output shows where the files are copied.

About this task

The following notes apply to specific databases:

- **Oracle 11g**
  - If you did not create the default database when you installed Oracle product, you must manually create the database before you run the createIdMgrPropExtDbTables script. The value of the *ORACLE_SID* variable is the same value as the name of the database.
  - If you want to create the tables in the schema that you specified using DBSCHEMA (described in the previous section, Specifying the database schema) ensure that you create the specified schema in this database before you run the createIdMgrPropExtDbTables script.
  - On the AIX, HP-UX, Linux, and Solaris operating systems, run the createIdMgrPropExtDbTables script either as an Oracle user or as a root user with database administrator (dba) rights and appropriate permissions to run SQL queries as a system database administrator (sysdba).
- **IBM DB2**
  - On the Windows operating systems, you must initialize the DB2 environment before you run the createIdMgrPropExtDbTables script. At the Windows command prompt, enter db2cmd to open a new DB2 command window and run the createIdMgrPropExtDbTables batch file from this prompt.
- **Microsoft SQL Server**
  - Open a command window, change to the *app_server_root*\bin directory, and enter the following commands to replace the variables in the SQL files. If WebSphere Application Server and the database are not on the same system, set the **SCHEMA_LOCATION** value to the location where you copied the SQL files.

    ```
    set SCHEMA_LOCATION=app_server_root\etc\wim\setup\lookaside
    set DBTYPE=sqlserver
    set SCHEMA_DEST_LOCATION=<location where the updated SQL files with replaced variables should be copied>
    set DBOWNER=dbo
    ws_ant.bat -f app_server_root\etc\wim\setup\filterbuild.xml
    ```

    Note: If SCHEMA_DEST_LOCATION is not set, the updated SQL files are copied to a directory with the name as the value not substituted under the current directory. The output shows where the files are copied.

The following default instance is created as a part of the database installation:

- DB2: *DB2*
- Informix: *demo_on*
- SQL Server: *%computername%*

The Informix database is created with the following environment:

```
CLIENT_LOCALE=EN_US.CP1252
DB_LOCALE=EN_US.8859-1
SERVER_LOCALE=EN_US.CP1252
DBLANG=EN_US.CP125
```

Procedure

Run the createIdMgrPropExtDbTables.sh script or createIdMgrPropExtDbTables.bat script to create the tables in the property extension repository.
Run the script from the following location or from the directory to which you previously copied the script file:

AIX, HP-UX, Linux, and Solaris operating systems

app_server_root/etc/wim/setup/bin/createIdMgrPropExtDbTables.sh

Windows

app_server_root\etc\wim\setup\bin\createIdMgrPropExtDbTables.bat

Use the following parameters to specify the values that you require when you run the script:

-b Use this parameter to specify the home directory of the database.

This value is a string value that is required for all database types.

-d Use this parameter to specify the schema of the database.

The value of this parameter should be the same value that you specified for DBSCHEMA (described in the previous section, Specifying the database schema).

This value is a string value that is optional for DB2, Derby, and SQL Server databases, if you want to specify the database schema where you want to create the federated repository tables. This value is not required for Oracle and Informix databases.

-h Use this parameter to display the help information. (Optional)

-i Use this parameter to specify the home directory of the database instance.

This value is a string value that is required for a DB2 database only; do not specify a value for other database types.

This parameter applies to the AIX, HP-UX, Linux, and Solaris operating systems.

-n Use this parameter to specify the name of the database to which you are connecting.

For an Oracle database, the value of the *ORACLE_SID* variable is the same as the name of the database.

This value is a string value that is required for all database types.

-p Use this parameter to specify the password of the database administrator.

This value is a string value that is required for DB2, Oracle, Informix, and SQL Server databases only; do not specify a value for a Derby database.

-s On AIX, HP-UX, Linux, and Solaris operating systems, this parameter specifies the location of the *app_server_root*/etc/wim/setup directory, or the location to which the updated files are copied according to the steps in the previous section, Specifying the database schema.

On Windows operating systems, this parameter specifies the location of the *app_server_root*\etc\wim\setup directory, or the location to which the updated files are copied according to the steps in the previous section, Specifying the database schema.

This value is a string value that is required for all database types.

-t Use this parameter to specify a database type.

- On the AIX, HP-UX, Linux, and Solaris operating systems, specify one of the following valid values: db2, oracle, informix, derby.
- On the Windows operating systems, specify one of the following valid values: db2, oracle, informix, derby, or sqlserver.

This value is a string value that is required for all database types.

-u Use this parameter to specify the user ID of the database administrator.

This value is a string value that is required for DB2, Oracle, Informix, and SQL Server databases only; do not specify a value for a Derby database.

Example

Run the appropriate script for your database and operating system to create tables in the property extension repository. Use the sample values to specify database parameters. If the database exists on a system where WebSphere Application Server is not installed, the following examples assume that your PATH variable includes an entry for the location to which you copied the script files. For the AIX, HP-UX, Linux, and Solaris operating systems, the entry might be the *app_server_root*/etc/wim/setup/bin/ or the /setup/bin/ directory. For Windows operating systems, the entry might be the *app_server_root*\etc\wim\setup\bin\ or the \setup\bin\ directory.

The examples in the following section are organized into multiple lines for illustration purposes only.

On the AIX, HP-UX, Linux, and Solaris operating systems:

Oracle databases

```
createIdMgrPropExtDbTables.sh
-b /space/oracle/product/10.2.0/Db_1/
-n orcl
-u system
-p manager
-s /opt/IBM/WebSphere/AppServer1/etc/wim/setup
-t oracle
```

Informix databases

```
createIdMgrPropExtDbTables.sh
-b /opt/IBM/informix/
-n demo_on
-u informix
-p informix
-s /opt/IBM/WebSphere/AppServer/etc/wim/setup
-t informix
```

DB2 databases

```
createIdMgrPropExtDbTables.sh
-b /opt/ibm/db2/V9.1/
-n db2inst1
-p db2inst1
-s /opt/IBM/WebSphere/AppServer/etc/wim/setup
-t DB2
-u db2inst1
-i /home/db2inst1/
```

Derby databases

```
createIdMgrPropExtDbTables.sh
-b /opt/ibm/derby/
-n test11
-s /opt/IBM/WebSphere/AppServer/etc/wim/setup
-t derby
```

On the Windows operating systems:

Oracle databases

```
createIdMgrPropExtDbTables.bat
-b "c:\oracle\product\10.2.0\Db_1"
-n orcl
-u system
-p manager
-s "c:\Program Files\IBM\WebSphere\AppServer1\etc\wim\setup"
-t oracle
```

Informix databases

```
createIdMgrPropExtDbTables.bat
-b "c:\Program Files\IBM\informix"
-n demo_on
```

```
-u informix
-p informix
-s "c:\Program Files\IBM\WebSphere\AppServer\etc\wim\setup"
-t informix
```

DB2 databases

```
createIdMgrPropExtDbTables.bat
-t db2
-u db2admin
-p sec001ret#
-n test23
-b "c:\Program Files\IBM\SQLLIB"
-s "c:\Program Files\IBM\WebSphere\AppServer1\etc\wim\setup"
```

Derby databases

```
createIdMgrPropExtDbTables.bat
-t derby
-b "c:\Derby"
-n test11
-s "c:\Program Files\IBM\WebSphere\AppServer1\etc\wim\setup"
```

Microsoft SQL Server databases

```
createIdMgrPropExtDbTables.bat
-t sqlserver
-u sa
-p sec001ret#
-n sqlsrv
-b "c:\Progra~1\Micros~1\90\Tools"
-s "C:\Progra~1\IBM\WebSphere\AppServer1\etc\wim\setup"
```

What to do next

Run the setupIdMgrPropertyExtensionRepositoryTables command with the **skipDBCreation** parameter set to true to populate the tables that are created. For more information, read about setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands.

Manually setting up the property extension repository for DB2 for iSeries or DB2 for z/OS:

Use this task to set up the property extension repository for DB2 for iSeries or DB2 for z/OS.

Before you begin

The information in this topic applies in the following scenarios:

- The application server and the database both exist on the IBM i operating system.
- The application server and the database both exist on the z/OS operating system.
- The application server exists on a distributed operating system, but the database exists on either the IBM i or z/OS operating system.

If you do not have WebSphere Application Server installed in the system on which you are setting up the database, copy the following files from a system where WebSphere Application Server is installed to the system on which you are setting up the database:

DB2 for iSeries

app_server_root/etc/wim/setup/lookaside/db2iseries/dbclean.sql
app_server_root/etc/wim/setup/lookaside/db2iseries/schema.sql
app_server_root/etc/wim/setup/lookaside/db2iseries/primarykeys.sql
app_server_root/etc/wim/setup/lookaside/db2iseries/indexes.sql
app_server_root/etc/wim/setup/lookaside/db2iseries/references.sql
app_server_root/etc/wim/setup/lookaside/keys.sql
app_server_root/etc/wim/setup/lookaside/bootstrap.sql

DB2 for z/OS

app_server_root/etc/wim/setup/lookaside/db2zos/dbclean.sql
app_server_root/etc/wim/setup/lookaside/db2zos/schema.sql
app_server_root/etc/wim/setup/lookaside/db2zos/primarykeys.sql
app_server_root/etc/wim/setup/lookaside/db2zos/indexes.sql
app_server_root/etc/wim/setup/lookaside/db2zos/references.sql
app_server_root/etc/wim/setup/lookaside/keys.sql
app_server_root/etc/wim/setup/lookaside/bootstrap.sql

About this task

For information about how to create a database and run SQL queries in DB2 for iSeries, see the DB2 Universal Database for iSeries in the IBM iSeries Information Center.

For information about how to create a database and run SQL queries in DB2 for z/OS, see the Information Management Software for z/OS Solutions Information Center.

Procedure

1. Open a command window.
2. Change to the *app_server_root*/bin directory
3. Enter the following commands to replace the variables in the SQL files:
   a. `export SCHEMA_LOCATION=`*app_server_root*`/etc/wim/setup/lookaside`

      Set the SCHEMA_LOCATION value to the location where you copied the SQL files if you do not have WebSphere Application Server installed on the same system on which you are setting up the database.

   b. `export DBTYPE=<`*db_type*`>`

      where the value of *<db_type>* is db2iseries or db2zos

   c. To specify the database schema where you want to create the federated repository tables use the DBSCHEMA command. If you want to use the default schema, which is typically the namespace of the current database user, do not specify the DBSCHEMA command.

      ```
      export DBSCHEMA=dbschemaname
      ```

   d. `export TSPREFIX=<`*tsprefix*`>`

      where *<tsprefix>* is the tablespace prefix. The maximum length allowed for this string is 3 characters.

   e. `export SCHEMA_DEST_LOCATION=<`*schema_dest_location*`>`

      where *<schema_dest_location>* is the location where the updated SQL files with replaced variables should be copied. If SCHEMA_DEST_LOCATION is not set, the updated SQL files are copied to a directory with the name as the unsubstituted value under the current directory. The output indicates where the files are copied to.

   f. `./ws_ant.sh -f `*app_server_root*`/etc/wim/setup/filterbuild.xml`
4. Start the DB2 server.
5. Create a database.
6. Run the SQL files, which were previously referenced, to create the tables for the property extension repository. If you are setting up the database on the same system on which the application server is installed, the files are located in the following locations:

   DB2 for iSeries

   app_server_root/etc/wim/setup/lookaside/db2iseries/dbclean.sql
   app_server_root/etc/wim/setup/lookaside/db2iseries/schema.sql
   app_server_root/etc/wim/setup/lookaside/db2iseries/primarykeys.sql
   app_server_root/etc/wim/setup/lookaside/db2iseries/indexes.sql
   app_server_root/etc/wim/setup/lookaside/db2iseries/references.sql
   app_server_root/etc/wim/setup/lookaside/keys.sql
   app_server_root/etc/wim/setup/lookaside/bootstrap.sql

DB2 for z/OS

app_server_root/etc/wim/setup/lookaside/db2zos/dbclean.sql
app_server_root/etc/wim/setup/lookaside/db2zos/schema.sql
app_server_root/etc/wim/setup/lookaside/db2zos/primarykeys.sql
app_server_root/etc/wim/setup/lookaside/db2zos/indexes.sql
app_server_root/etc/wim/setup/lookaside/db2zos/references.sql
app_server_root/etc/wim/setup/lookaside/keys.sql
app_server_root/etc/wim/setup/lookaside/bootstrap.sql

Otherwise, run the SQL files from the location to which you copied the files. If you executed the commands to substitute variables according to the steps in the previous section, Specifying the database schema, the SQL files are copied to the location you specified for SCHEMA_DEST_LOCATION. If SCHEMA_DEST_LOCATION is not set, the updated SQL files are copied to a directory with the name as the unsubstituted value under the current directory. The output shows where the files are copied.

What to do next

Run the setupIdMgrPropertyExtensionRepositoryTables command with the **skipDBCreation** parameter set to true to populate the tables that are created. For more information, read about setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands.

Configuring the WebSphere Application Server data source:

Installed applications use data sources as resources to obtain connection to relational databases. To create these connections between an application and a relational database, WebSphere Application Server uses the driver implementation classes that are encapsulated by the JDBC provider, which is an object that represents vendor-specific JDBC driver classes to WebSphere Application Server. For access to a relational databases, applications use the JDBC drivers and data sources that you configure for WebSphere Application Server.

Procedure

1. Start the WebSphere Application Server administrative console.
2. Click **Security -> Global security**.
3. On the Configuration panel, expand Java Authentication and Authorization Service and click **J2C authentication data**.
4. Click **New** and enter the Alias, User ID and Password.
5. Click **Ok**.
6. On the WebSphere Application Server administrative console, expand **Resources**. Expand **JDBC** then click **JDBC Providers**.
7. In the Scope section, choose the **Node level** from the drop-down list.
8. Click **New** to create a new JDBC driver.
9. Select, in this order, the Database type, Provider type, Implementation type and Name. The Name automatically fills based on the implementation type you choose.
10. Click **Next** and configure the database class path. Click **Next**.
11. On the Summary page, click **Finish**.
12. Click **Save** to save your selections. The JDBC providers page then appears.
13. On the WebSphere Application Server administrative console, click **Data sources**.
14. Click **New** to create a new data source. Enter the Data source name and the JNDI name, and choose the authentication alias from the drop-down list in Component-managed authentication alias. The JNDI name should match the datasourceName value set in `wimconfig.xml`. By default, it is `jdbc/wimDS`.

Note: For Apache Derby v10.2 embedded databases, leave the Component-managed authentication alias field set to `NONE`.

15. Click **Next**.
16. Enter the Database name and deselect the checkbox, **Use this data source in container managed persistence (CMP)**. Click **Next**.
17. On the Summary page, click **Finish**.
18. The Data sources page displays. Click **Save**, Then select the check box for the authentication alias previously created. Click **Test Connection**. The message should indicate that the connection is successful. Ignore any warnings, and then click **Next**.
19. Save the configurations, and restart WebSphere Application Server.

Configuring an entry mapping repository in a federated repository configuration:

Follow this task to configure an entry mapping repository that is used to store data for managing profiles on multiple repositories.

About this task

An *entry-level join* means that the federated repository configuration uses multiple repositories simultaneously and recognizes the entries in the different repositories as entries representing distinct entities. For example, a company might have a Lightweight Directory Access Protocol (LDAP) directory that contains entries for its employees and a database that contains entries for business partners and customers. By configuring an entry mapping repository, a federated repository configuration can use both the LDAP and the database at the same time. The federated repository configuration hierarchy and constraints for identifiers provide the aggregated namespace for both of those repositories and prevent identifiers from colliding.

When you configure an entry mapping repository, you can supply a valid data source, a direct connection configuration, or both. The system first tries to connect by way of the data source. If the data source is not available, then the system uses the direct access configuration.

Restriction: You cannot configure an entry mapping repository in a mixed-version deployment manager cell.

Procedure

1. Configure the WebSphere Application Server data source. See “Configuring the WebSphere Application Server data source” on page 1373.
2. Set up the entry mapping repository using wsadmin. See “Setting up an entry mapping repository, a property extension repository, or a custom registry database repository using wsadmin commands” on page 1359; ignore the "Before you begin" options.
3. Configure the entry mapping repository into the federated repository by doing the following:
   a. In the administrative console, click **Security > Global security**.
   b. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
   c. Click **Entry mapping repository**.
   d. Supply the name of the data source in the Data source name field.
   e. Select the type of database that is used for the property extension repository.
   f. Supply the name of the Java database connectivity (JDBC) driver in the JDBC driver field.
      Values include:

      DB2 `com.ibm.db2.jcc.DB2Driver`

DB2 for iSeries
com.ibm.db2.jcc.DB2Driver

Informix
com.informix.jdbc.IfxDriver

DataDirect Connect
com.ddtek.jdbc.sqlserver.SQLServerDriver

Derby org.apache.derby.jdbc.EmbeddedDriver

Microsoft SQL Server
com.microsoft.sqlserver.jdbc.SQLServerDriver

Oracle
oracle.jdbc.driver.OracleDriver

g. Supply the database URL that is used to access the property extension repository with JDBC in the Database URL field. Use an alphanumeric text string that conforms to the standard JDBC URL syntax.

Values include:

DB2 jdbc:db2:wim

Informix
jdbc:informix-sqli://*host_name*:1526/wim:INFORMIXSERVER=*IFXServerName*;

DataDirect Connect
jdbc:datadirect:sqlserver://*host_name*:1433;databaseName=wim;selectMethod=cursor;

Derby jdbc:derby:c:\derby\wim

Microsoft SQL Server
jdbc:sqlserver://*host_name*:1433;databaseName=wim;selectMethod=cursor;

Oracle
jdbc:oracle:thin:@*host_name*:*port*:*dbname*

h. Supply the user name of the database administrator in the Database administrator user name field.

i. Supply the password of the database administrator in the Password field.

j. Click **OK**.

Results

After completing these steps, your federated repository configuration, which includes an entry mapping repository, is configured.

What to do next

1. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
2. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
3. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Entry mapping repository settings:

Use this page to configure an entry mapping repository that is used to store data for managing profiles on multiple repositories.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Additional properties, click **Entry mapping repository**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Data source name:

Specifies the Java Naming and Directory Interface (JNDI) name of the data source that is used to access the entry mapping repository.

Default: `jdbc/wimDS`

Database type:

Specifies the type of database that is used to access the entry mapping repository.

Default: `DB2`

JDBC driver:

Specifies the Java Database Connectivity (JDBC) driver that is used to access the entry mapping repository.

Values include:

DB2 `com.ibm.db2.jcc.DB2Driver`

DB2 for iSeries
`com.ibm.db2.jcc.DB2Driver`

DataDirect Connect
`com.ddtek.jdbc.sqlserver.SQLServerDriver`

Informix
`com.informix.jdbc.IfxDriver`

Oracle
`oracle.jdbc.driver.OracleDriver`

Microsoft SQL Server
`com.microsoft.sqlserver.jdbc.SQLServerDriver`

Derby `org.apache.derby.jdbc.EmbeddedDriver`

Database URL:

Specifies the web address for the entry mapping repository.

Values include:

DB2 `jdbc:db2:wim`

Derby `jdbc:derby:c:\derby\wim`

DataDirect Connect

`datadirect:sqlserver://:host_name1433;databaseName=wim;selectMethod=cursor;`

Oracle

`jdbc:oracle:thin:@host_name:port:dbname`

Microsoft SQL Server

`jdbc:sqlserver://host_name:1433;databaseName=wim;selectMethod=cursor;`

Informix

`jdbc:informix-sqli://host_name:port/wim:INFORMIXSERVER=IFXServerName;`

Database administrator user name:

Specifies the user name of the database administrator that is used to access the entry mapping repository.

Password:

Specifies the password that is used to enable the database administrator to access the entry mapping repository.

Configuring supported entity types in a federated repository configuration:

Follow this task to configure supported entity types for user and group management.

About this task

You must configure the supported entity types before you can manage this account with Users and Groups in the administrative console. The supported entity types are Group, OrgContainer, and PersonAccount. A Group entity represents a simple collection of entities that might not have any relational context. An OrgContainer entity represents an organization, such as a company or an enterprise, a subsidiary, or an organizational unit, such as a division, a location, or a department. A PersonAccount entity represents a human being. You cannot add or delete the supported entity types, because these types are predefined.

The Base entry for the default parent determines the repository location where entities of the specified type are placed on write operations by user and group management.

Note: To manage users and groups, click **Users and Groups** in the console navigation tree. Click either **Manage Users** or **Manage Groups**.

Note: You must restart the server or dmgr if the federated repository has changed before using the Manage Users option. Otherwise, user or group changes made to the repository could be lost after restart.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Click **Supported entity types** to view a list of predefined entity types.
4. Click the name of a predefined entity type to change its configuration.
5. Supply the distinguished name of a base entry in the repository in the Base entry for the default parent field. This entry determines the default location in the repository where entities of this type are placed on write operations by user and group management.

6. Supply the relative distinguished name (RDN®) properties for the specified entity type in the Relative Distinguished Name properties field. Possible values are `cn` for Group, `uid` or `cn` for PersonAccount, and `o`, `ou`, `dc`, and `cn` for OrgContainer. Delimit multiple properties for the OrgContainer entity with a semicolon (;).

   The following list outlines known requirements and limitations that apply to specific Lightweight Directory Access Protocol (LDAP) servers:

   Using Microsoft Active Directory as the LDAP server

   - Unless you modify the LDAP schema to use `uid`, you must specify `cn` in the Relative Distinguished Name (RDN) properties field for the PersonAccount entity type.
   - Secure Sockets Layer communications must be enabled to create users with passwords. To select the **Require SSL communications** option, see the topic "Configuring Lightweight Directory Access Protocol in a federated repository configuration" on page 1339.
   - Typically the value of `user` is specified as the value in the Object classes field for the PersonAccount entity type and the value of `group` is specified as the value in the Object classes field for the Group entity type.

   Using a Lotus Domino Enterprise Server as the LDAP server

   - Typically, the value of `cn` is specified in the Relative Distinguished Name (RDN) properties field for the PersonAccount entity type. The value of `uid` is also acceptable.
   - Typically, both `inetOrgPerson` and `dominoPerson` are used as values in the Object classes field for the PersonAccount entity type.

   Using Sun ONE Directory Server as the LDAP server

   - Typically, `groupOfUniqueNames` is specified as the value in the Object classes field for the Group entity type.
7. Click **OK**.

Results

After completing these steps, your federated repository configuration, which uses supported entity types, is configured.

What to do next

1. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
2. If you are enabling security, complete the remaining steps as specified in "Enabling security for the realm" on page 1186. As the final step, validate this setup by clicking **Apply** on the Global security panel.
3. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Supported entity types collection:

Use this page to list entity types that are supported by the member repositories or to select an entity type to view or change its configuration properties.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.

2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Additional properties, click **Supported entity types**.

You must configure the supported entity types before you can manage this account with Users and Groups in the administrative console. The Base entry for the default parent determines the repository location where entities of the specified type are placed on write operations by user and group management.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Entity type:

Specifies the entity type name.

Base entry for the default parent:

Specifies the distinguished name of a base entry in the repository.

This entry determines the default location in the repository where entities of this type are placed on write operations by user and group management.

Relative Distinguished Name properties:

Specifies the relative distinguished name (RDN) properties for the specified entity type.

Possible values are `cn` for Group, `uid` or `cn` for PersonAccount, and `o`, `ou`, `dc`, and `cn` for OrgContainer. Delimit multiple properties for the OrgContainer entity with a semicolon (;).

Supported entity types settings:

Use this page to configure entity types that are supported by the member repositories.

To view this administrative console page, complete the following steps:
1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Additional properties, click **Supported entity types**.
4. Click the name of a configured entity type to view or change its configuration.

You must configure the supported entity types before you can manage this account with Users and Groups in the administrative console. The Base entry for the default parent determines the repository location where entities of the specified type are placed on write operations by user and group management.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Entity type:

Specifies the name of the entity type.

Base entry for the default parent:

Specifies the distinguished name of a base entry in the repository.

This entry determines the default location in the repository where entities of this type are placed on write operations by user and group management.

Relative Distinguished Name properties:

Specifies the relative distinguished name (RDN) properties for the specified entity type.

Possible values are `cn` for Group, `uid` or `cn` for PersonAccount, and `o`, `ou`, `dc`, and `cn` for OrgContainer. Delimit multiple properties for the OrgContainer entity with a semicolon (;).

Managing repositories in a federated repository configuration:

Follow this topic to manage repositories in a federated repository configuration.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**. Repositories that are configured in the system are listed in the collection panel. This list includes repositories that are configured using the federated repository functionality as well as repositories that are created using wsadmin commands described in the topic IdMgrRepositoryConfig command group for the AdminTask object.
4. Optional: Click **Add** to configure a new external repository. The Lightweight Directory Access Protocol (LDAP) repository configuration settings are described in detail in “Configuring Lightweight Directory Access Protocol in a federated repository configuration” on page 1339.

   Restriction: You cannot add a database repository using the administrative console. This repository configuration is supported by using wsadmin commands only.
5. Optional: Click **Delete** to delete a repository that you specified previously using the administrative console or wsadmin commands.

   Restriction: You cannot delete the built-in, file-based repository from the collection panel.
6. Optional: Select one of the LDAP repository identifier entries to view or update an external repository that is configured in the system previously. The steps to configure LDAP settings are described in detail in “Configuring Lightweight Directory Access Protocol in a federated repository configuration” on page 1339.

   Restriction: While database repositories that are configured in the system are listed in the collection panel, you cannot update a database repository using the administrative console. Updates to a database repository are supported by using wsadmin commands only.
7. Click **OK**.

Results

After completing these steps, the collection panel under Managing repositories reflects a current list of repositories that are configured in your system.

What to do next

1. To add one or more external repositories that are listed on this collection panel into the realm, see “Managing the realm in a federated repository configuration” on page 1316.
2. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply**

on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.

3. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
4. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Replicating changes to a built-in, file-based repository:

Changes to built-in, file-based repositories are not automatically replicated to managed nodes in a federated repositories configuration. You need to use the administrative console to replicate the changes you make to a built-in, file-based repository.

About this task

The network deployment support in a federated repositories configuration only updates the in-memory state of the processes that are running on the managed nodes. Because WebSphere Application Server synchronizes the file systems, the network deployment support does not attempt to update the file systems of the managed nodes.

You must synchronize the node configuration to replicate the changes to the built-in, file-based repository.

Procedure

1. In the administrative console, click **System Administration > Nodes**. to access the nodes panel.
2. On the Nodes panel, select all the relevant nodes for which the changes to the built-in, file-based repository need to be made.
3. Click **Full Resynchronize**. The resynchronize operation resolves conflicts among configuration files and can take several minutes to complete.

Results

After completing these steps, your federated repository configuration of managed nodes reflects the changes to the built-in, file-based repository.

Manage repositories collection:

Use this page to list repositories that are configured in the system or to select a repository to view or change its configuration properties. You can add or delete external repositories.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Repository identifier:

Specifies a unique identifier for the repository. This identifier uniquely identifies the repository within the cell.

Repository type:

Specifies the repository type, such as File or LDAP.

Add:

Select to add a new LDAP, custom or file repository.

Repository reference settings:

Use this page to configure a repository reference. A repository reference is a single repository that contains a set of identity entries that are referenced by a base entry into the directory information tree.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Click **Add base entry to realm**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Repository:

Specifies a unique identifier for the repository. This identifier uniquely identifies the repository within the cell.

Expand the drop-down list to display a list of previously defined repository identifiers.

Distinguished name that uniquely identifies this set of entries in the realm:

Specifies the distinguished name (DN) that uniquely identifies this set of entries in the realm.

If multiple repositories are included in the realm, it is necessary to define an additional distinguished name that uniquely identifies this set of entries within the realm. Overlapping base entries are not supported. You should not define two base entries where one is c=us, and the other is o=myorg,c=us in the same realm; otherwise a search returns duplicate results.

Distinguished name of a base entry in this repository:

Specifies the Lightweight Directory Access Protocol (LDAP) distinguished name (DN) of the base entry within the repository. The entry and its descendents are mapped to the subtree that is identified by the unique base name entry field.

If this field is left blank, then the subtree defaults to the root of the LDAP repository.

Increasing the performance of the federated repository configuration:

Follow this page to manage the realm in a federated repository configuration.

Before you begin

The settings that are available on the Performance panel are independent options that pertain specifically to the federated repositories functionality. These options do not affect your entire WebSphere Application Server configuration.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories** > *repository_name*.
4. Under Additional properties, click **Performance**.
5. Optional: Select the **Limit search time** option and enter the maximum number of milliseconds that the Application Server can use to search through your Lightweight Directory Access Protocol (LDAP) entries.
6. Optional: Select the **Limit search returns** option and enter the maximum number of entries to return that match the search criteria.
7. Optional: Select the **Use connection pooling** option to specify whether the Application Server can store separate connections to the LDAP server for reuse.
8. Optional: Select the **Enable context pool** option to specify whether multiple applications can use the same connection to the LDAP server. If you select the option, specify the initial, preferred, and maximum number of entries that can use the same connection. The **Enable context pool** option can be enabled either in conjunction with the **Use connection pool** option or separately. If this option is disabled, a new connection is created for each context. You can also select the **Context pool times out** option and specify the number of seconds after which the entries in the context pool expire.
9. Optional: Set the **Maximum size** value of the context pool to zero (0).
10. Optional: Select the **Cache the attributes** option and specify the maximum number of search attribute entries. This option enables WebSphere Application Server to save the LDAP entries so that it can search the entries locally rather than making multiple calls to the LDAP server. Click the **Cache times out** option that is associated with the **Cache the attributes** option to specify the maximum number of seconds that the Application Server can save these entries.
11. Optional: Select the **Cache the search results** option and specify the maximum number of search result entries. This option enables WebSphere Application Server to save the results of a search inquiry instead of making multiple calls to the LDAP server to search and retrieve the results of that search. Click the **Cache times out** option that is associated with the **Cache the search results** option to specify the maximum number of seconds that the Application Server can save the results.
12. Optional: Create the root DataObject object locally using the com.ibm.websphere.wim.util.SDOHelper.createRootDataObject method instead of the com.ibm.websphere.wim.ServiceProvider.createRootDataObject method.

Results

These options are available to potentially increase the performance of your federated repositories configuration. However, the any increase in performance is dependant upon your specific configuration.

Lightweight Directory Access Protocol performance settings:

Use this page to minimize impacts to performance by adding opened connections and contexts to internally maintained pools and reusing them. Also minimize performance impacts by maintaining internal caches of retrieved data.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **Performance**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Limit search time:

Specifies the timeout value in milliseconds for a Lightweight Directory Access Protocol (LDAP) server to respond before stopping a request.

Data type:	Integer
Units:	Milliseconds
Default:	0
Range:	Equal to or greater than 0. A value of 0 specifies that no search time limit exists.

Limit search returns:

Specifies the maximum number of entries that are returned in a search result.

Data type:	Integer
Units:	Entries
Default:	0
Range:	Equal to or greater than 0. A value of 0 specifies that no search return limit exists.

Use connection pooling:

Specifies whether to utilize the connection pooling function, which is provided in the Software Development Kit (SDK).

Connection pooling is maintained by the Java run time. It is configured by system properties.

Default:	Disabled
Range:	Enabled or Disabled

Enable context pool:

Specifies whether context pooling is enabled to the LDAP server. To improve performance, use the context pool in combination with connection pooling.

Default:	Enabled
Range:	Enabled or Disabled

Initial size:

Specifies the number of context instances in the pool when the pool is initially created by the LDAP repository.

Data type:	Integer
Default:	1
Range:	1 to 50

Preferred size:

Specifies the preferred number of context instances that the context pool maintains. Both in-use and idle context instances contribute to this number.

Data type: Integer
Default: 3
Range: 0 to 100

Maximum size:

Specifies the maximum number of context instances that can be maintained concurrently by the context pool. Both in-use and idle context instances contribute to this number.

When the pool size reaches the maximum size, no new context instances can be created for a new request. The new request is blocked until a context instance is released or removed. The request periodically checks for context instances that are available in the pool. A request for a pooled context instance uses an existing pooled and idle context instance or a newly created pooled context instance.

A maximum pool size of 0 indicates that the context pool can maintain an infinite number of context instances.

Data type: Integer
Default: 0

Context pool times out:

Specifies the number of seconds for the context pool to time out and remove idle context instances.

A timeout value of 0 indicates that the context pool does not time out context instances.

Data type: Integer
Default: 0

Distribution policy:

Specifies the distribution policy for the cache in a clustered environment, which is one of the following:

None Sends out new entries, both ID and data, and updates to those entries.

Push Requests data from other servers in the cluster when that data is not locally present.

Push and pull
Sends out IDs for new entries and requests from other servers in the cluster entries for IDs that were previously broadcast. The dynamic cache always sends out cache entry invalidations.

Cache the attributes:

Specifies whether to cache the attributes that are returned from the LDAP server.

Default: Enabled
Range: Enabled or Disabled

Cache size:

Specifies the maximum size of the cache.

Data type: Integer
Default: 4000
Range: Equal to or greater than 100

Cache times out:

Specifies the maximum number of seconds that the cached search results can stay in the cache.

A timeout value of 0 indicates that the cached search results stay in the cache until update operations are made.

Data type: Integer
Units: Seconds
Default: 1200
Range: Equal to or greater than 0

Cache the search results:

Specifies whether to cache the search results that are returned from the LDAP server.

Default: Enabled
Range: Enabled or Disabled

Cache size:

Specifies the maximum size of the cache.

Data type: Integer
Default: 2000
Range: Equal to or greater than 100

Cache times out:

Specifies the maximum number of seconds that the cached search results can stay in the cache.

A timeout value of 0 indicates that the cached search results stay in the cache until update operations are made.

Data type: Integer
Units: Seconds
Default: 600
Range: Equal to or greater than 0

Using custom adapters for federated repositories:

When the custom adapters for federated repositories are part of the default realm, the users and groups can be managed using wsadmin commands or the administrative console.

About this task

If custom adapters for federated repositories are part of the default realm, you use the administrative console to manage the users and groups in the realm.

Note: The default parent for PersonAccount and Group entities needs to be the same as the base entry of the custom adapter.

To view this administrative console page, complete the following steps:

- In the administrative console, click **Security > Global security**.
- Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
- Under Additional properties, click **Supported entity types**.

You must configure the supported entity types before you can manage this account with Users and Groups in the administrative console. The Base entry for the default parent determines the repository location where entities of the specified type are placed on write operations by user and group management.

Procedure

1. In the administrative console, click **Users and Groups** to access users and groups panel.
2. Click **Manage Groups** to test the basic functions of the custom adapter with respect to custom adapters for federated repositories.
3. Click **Manage Users** to test the basic functions of the custom adapter with respect to custom adapters for federated repositories.

   Note: You must restart the server or dmgr if the federated repository has changed before using the Manage Users option. Otherwise, user or group changes made to the repository could be lost after restart.

Results

After completing these steps, you will have ensured that the custom adapter is being used properly.

What to do next

Adjustments to the custom adapter can be made by using the wsadmin tool to make configuration changes. See “Configuring custom adapters for federated repositories using wsadmin” on page 1389 for more details.

Sample custom adapters for federated repositories examples:

Out of the box adapters for federated repositories provide File, LDAP, and Database adapters for your use. These adapters implement the com.ibm.wsspi.wim.Repository software programming interface (SPI). A virtual member manager custom adapter needs to implement the same SPI.

Developing custom adapters for federated repositories

Out of the box adapters for federated repositories provide File, LDAP and Database adapters for your use. All these adapters implement the `com.ibm.wsspi.wim.Repository` SPI. See the `com.ibm.wsspi.wim.Repository` SPI for more information. As you develop a virtual member manager custom adapter, you need to implement the same SPI.

Custom adapters for federated repositories must not depend on any WebSphere Application Server components, such as data sources and enterprise beans. These WebSphere Application Server components require that security is initialized and enabled prior to startup. If your implementation of custom adapters for federated repositories needs to use data sources to connect to a database, you need to use Java database connectivity (JDBC) to make the connection during server startup. Then, at a later time, switch to using the data sources when the data source is available.

There are examples of suggested behavior and requirements of custom adapters for federated repositories that you can find in the sample code.

A sample custom adapter for federated repositories

A sample custom adapter implementation has been provided as an example. The custom adapter is based on file repository. The sample source code and class files are bundled in vmmsampleadapter.jar. The `vmmsampleadapter.jar` can be downloaded at this location: http://www.ibm.com/developerworks/websphere/downloads/samples/vmmsampleadapter.html.

Contents of the vmmsampleadapter.jar file are as follows:

- Class files for the sample adapter:
  - `com/ibm/ws/wim/adapter/sample/AbstractAdapterImpl.class`
  - `com/ibm/ws/wim/adapter/sample/SampleFileAdapter.class`
  - `com/ibm/ws/wim/adapter/sample/XPathHelper.class`
- Source code for the sample adapter:
  - `src/com/ibm/ws/wim/adapter/sample/AbstractAdapterImpl.java`
  - `src/com/ibm/ws/wim/adapter/sample/SampleFileAdapter.java`
  - `src/com/ibm/ws/wim/adapter/sample/XPathHelper.java`

Note: The sample files should not be used in the production environment. You should make a copy of these files, rename them, and update them based on your specific adapter implementation. Refer to the Javadoc in the source code for more information.

com/ibm/ws/wim/sample/adapter/AbstractAdapterImpl.java
Provides an abstract implementation class which handles most of the repository independent internal operations for the adapter and defines some simple abstract methods that should be implemented by the custom adapter. For most cases, you may not need to change this file.

com/ibm/ws/wim/sample/adapter/SampleFileAdapter.java
Extends from the AbstractAdapterImpl class and implements the abstracts method. This class implements the abstract methods using file as the repository. Adapter providers can use this class as a reference to implement these methods specific to their adapters.

com/ibm/ws/wim/sample/adapter/XPathHelper.java
Defines a helper class to parse the XPath search expression and build the search tree. This helper class also contains the method to evaluate the search expression. If your repository supports a search expression, then you need to convert XPath expression to an expression that your repository can process and let your repository evaluate the expression. This helper class evaluates the search expression based on the use of dataobjects. You can overwrite the `evaluate()` method to perform the evaluation using other objects, such as **java.util.Map**.

Some utility classes have been provided to help adapter providers. Most of these utility methods are used in the sample adapter. Refer to http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.javadoc.doc/vmm/com/ibm/wsspi/wim/package-summary.html for more details.

Establishing custom adapters for federated repositories:

Out of the box adapters for federated repositories provide File, LDAP, and Database adapter for your use. These adapters implement the com.ibm.wsspi.wim.Repository software programming interface (SPI). Custom adapters for federated repositories need to implement the same SPI.

Before you begin

Refer to the Repository SPI implementation information in the related references for information about the custom adapters for federated repositories SPI.

Refer to the sample custom adapter code that is available in the vmmsampleadapter.jar file. The JAR file contains the sample customer adapter code in the src/ directory. The vmmsampleadapter.jar can be downloaded at this location: http://www.ibm.com/developerworks/websphere/library/samples/vmmsampleadapter.html

Note:

- The sample that is provided is intended to familiarize you with the features of custom adapters for federated repositories and the handling of various types of dataobjects. Do not use this sample in an actual production environment.
- Copy the AbstractAdapterImpl class and rename it before making changes. Make sure that the new name is appropriate for your adapter.

Custom adapters for federated repositories must not depend on any WebSphere Application Server components, such as data sources and enterprise beans. These WebSphere Application Server components require that security is initialized and enabled prior to startup. If your implementation of the virtual member manager custom adapter needs to use data sources to connect to a database, you need to use Java database connectivity (JDBC) to make the connection during server startup. Then, at a later time, switch to using the data sources when the data source is available.

Procedure

1. Build your implementation.

   Note: EMF JAR files contain version number in their names, such as v200607270021. Make sure to change the version number to reflect your installation.

   To compile your code, you need the following JAR files in the classpath:

   - *app_server_root*/plugins/com.ibm.ws.runtime_6.1.0.jar
   - *app_server_root*/plugins/org.eclipse.emf.commonj.sdo_2.1.0.v200607270021.jar
   - *app_server_root*/plugins/org.eclipse.emf.ecore_2.2.1.v200607270021.jar
   - *app_server_root*/plugins/org.eclipse.emf.common_2.2.1.v200607270021.jar
   - *app_server_root*/plugins/org.eclipse.emf.ecore.xmi_2.2.0.v200607270021.jar
   - *app_server_root*/plugins/org.eclipse.emf.ecore.sdo_2.2.0.v200607270021.jar

   Here is an example:

   ```
   "${java.home}/bin/javac -classpath
   app_server_root/plugins/com.ibm.ws.runtime_6.1.0.jar;app_server_root/plugins/org.eclipse.emf.commonj.sdo_2.1.0.
   v200607270021.jar;app_server_root/plugins/org.eclipse.emf.ecore_2.2.1.v200607270021.jar;
   app_server_root/plugins/org.eclipse.emf.
   common_2.2.1.v200607270021.jar;app_server_root/plugins/org.eclipse.emf.ecore.xmi_2.2.0.v200607270021.jar;
   app_server_root/plugins/org.eclipse.emf.
   ecore.sdo_2.2.0.v200607270021.jar your_implementation_file.java"
   ```

2. Copy the generated class files or the packaged JAR file to the product classpath. The preferred location is the *app_server_root*/lib/ext directory. This should be copied to the classpaths of all the product processes (cell and all NodeAgents).
3. Configure your custom adapter by following the steps in “Configuring custom adapters for federated repositories using wsadmin.”
4. Test your custom adapter by following the steps in “Using custom adapters for federated repositories” on page 1386

What to do next

“Configuring custom adapters for federated repositories using wsadmin” provides details about configuring your custom adapter with the wsadmin tool.

Configuring custom adapters for federated repositories using wsadmin:

You can use the Jython or Jacl scripting language with the wsadmin tool to define custom adapters in the federated repositories configuration file.

Before you begin

Shut down the WebSphere Application Server and the wsadmin command window.

About this task

The federated repositories configuration file, `wimconfig.xml`, is shipped with WebSphere Application Server 6.1.x and is located in the *app_server_root*/`profiles/`*profile_name*`/config/cells/`*cell_name*`/wim/config` directory.

Note: For additional information about the commands to use for this topic, see the IdMgrRepositoryConfig command group for the AdminTask object topic.

Use the following steps to add a custom adapter to any federated repositories configuration file and to any realm defined within the configuration file.

Procedure

1. Open the `wimconfig.xml` file with a text editor.
2. Add a new config:repositories element to the file. This element should be placed before the config:realmConfiguration element.

   The following example configures a custom repository to use the **com.ibm.ws.wim.adapter.sample.SampleFileAdapter** class and sets the **SampleFileRepository** repository as the identifier:

```
<config:repositories adapterClassName="com.ibm.ws.wim.adapter.sample.SampleFileAdapter"
id="SampleFileRepository"/>
```

3. Save the `wimconfig.xml` file and close the text editor.
4. Copy the `vmmsampleadapter.jar` file that is provided to `app_server_root/lib`.
5. Enter the following command to start the wsadmin tool:

```
wsadmin –conntype none
```

6. Disable paging in the common repository configuration. Set the supportPaging parameter for the updateIdMgrRepository command to false to disable paging.

   Note: You must perform this step because the sample adapter does not support paging.

   The following examples use the **SampleFileRepository** repository as the identifier for the custom repository.

   Using Jython:

```
AdminTask.updateIdMgrRepository('-id SampleFileRepository -supportPaging false')
```

   Using Jacl:

```
$AdminTask updateIdMgrRepository {-id SampleFileRepository -supportPaging
false}
```

   Note: A warning will appear until the configuration of the sample repository is complete.

7. Add the necessary custom properties for the adapter. Use the setIdMgrCustomProperty command repeatedly to add multiple properties. Use this command once per property to add multiple properties to your configuration. You must use both the **name** and **value** parameters to add the custom property for the specified repository. For example, to add a custom property of **fileName**, enter the following command.

   Using Jython:

```
AdminTask.setIdMgrCustomProperty('-id SampleFileRepository -name fileName
-value "c:\sampleFileRegistry.xml"')
```

   Using Jacl:

```
$AdminTask setIdMgrCustomProperty {-id SampleFileRepository -name fileName
-value "c:\sampleFileRegistry.xml"}
```

8. Add a base entry to the adapter configuration. Use the addIdMgrRepositoryBaseEntry command to specify the name of the base entry for the specified repository. For example:

   Using Jython:

```
AdminTask.addIdMgrRepositoryBaseEntry('-id SampleFileRepository -name
o=sampleFileRepository')
```

   Using Jacl:

```
$AdminTask addIdMgrRepositoryBaseEntry {-id SampleFileRepository -name
o=sampleFileRepository}
```

9. Use the addIdMgrRealmBaseEntry command to add the base entry to the realm, which will link the realm with the repository:

   Using Jython:

```
AdminTask.addIdMgrRealmBaseEntry('-name defaultWIMFileBasedRealm -baseEntry o=sampleFileRepository')
```

   Using Jacl:

```
$AdminTask addIdMgrRealmBaseEntry {-name defaultWIMFileBasedRealm -baseEntry o=sampleFileRepository}
```

10. Save your configuration changes. Enter the following commands to save the new configuration and close the wsadmin tool.

    Using Jython:

```
AdminConfig.save()
exit
```

    Using Jacl:

```
$AdminConfig save
exit
```

    The following example displays the complete text of the newly-revised `wimconfig.xml` file:

```
<!--
  Begin Copyright

  Licensed Materials - Property of IBM

  virtual member manager

  (C) Copyright IBM Corp. 2005 All Rights Reserved.

  US Government Users Restricted Rights - Use, duplication or
  disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

  End Copyright
-->
<sdo:datagraph xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:config="http://www.ibm.com/websphere/wim
/config" xmlns:sdo="commonj.sdo">
  <config:configurationProvider maxPagingResults="500" maxSearchResults="4500"
maxTotalPagingResults="1000"
pagedCacheTimeOut="900" pagingEntityObject="true" searchTimeOut="600000">
    <config:dynamicModel xsdFileName="wimdatagraph.xsd"/>
    <config:supportedEntityTypes defaultParent="o=defaultWIMFileBasedRealm" name="Group">
      <config:rdnProperties>cn</config:rdnProperties>
    </config:supportedEntityTypes>
    <config:supportedEntityTypes defaultParent="o=defaultWIMFileBasedRealm" name="OrgContainer">
      <config:rdnProperties>o</config:rdnProperties>
      <config:rdnProperties>ou</config:rdnProperties>
      <config:rdnProperties>dc</config:rdnProperties>
      <config:rdnProperties>cn</config:rdnProperties>
    </config:supportedEntityTypes>
    <config:supportedEntityTypes defaultParent="o=defaultWIMFileBasedRealm" name="PersonAccount">
      <config:rdnProperties>uid</config:rdnProperties>
    </config:supportedEntityTypes>
    <config:repositories xsi:type="config:FileRepositoryType" adapterClassName="com.ibm.
    ws.wim.adapter.file.was.FileAdapter"
id="InternalFileRepository" supportPaging="false" supportSorting="false" messageDigestAlgorithm="SHA-1">
      <config:baseEntries name="o=defaultWIMFileBasedRealm"/>
    </config:repositories>
    <config:repositories adapterClassName="com.ibm.ws.wim.adapter.sample.SampleFileAdapter"
    id="SampleFileRepository">
      <config:CustomProperties name="fileName" value="c:\sampleFileRegistry.xml"/>
      <config:baseEntries name="o=sampleFileRepository"/>
    </config:repositories>
    <config:realmConfiguration defaultRealm="defaultWIMFileBasedRealm">
      <config:realms delimiter="@" name="defaultWIMFileBasedRealm" securityUse="active">
        <config:participatingBaseEntries name="o=defaultWIMFileBasedRealm"/>
        <config:participatingBaseEntries name="o=sampleFileRepository"/>
        <config:uniqueUserIdMapping propertyForInput="uniqueName" propertyForOutput="uniqueName"/>
        <config:userSecurityNameMapping propertyForInput="principalName" propertyForOutput="principalName"/>
```

```
        <config:userDisplayNameMapping propertyForInput="principalName" propertyForOutput="principalName"/>
        <config:uniqueGroupIdMapping propertyForInput="uniqueName" propertyForOutput="uniqueName"/>
        <config:groupSecurityNameMapping propertyForInput="cn" propertyForOutput="cn"/>
        <config:groupDisplayNameMapping propertyForInput="cn" propertyForOutput="cn"/>
      </config:realms>
    </config:realmConfiguration>
</config:configurationProvider></sdo:datagraph>
```

11. Restart the application server.

Configuring the user registry bridge for federated repositories using wsadmin scripting:

The user registry bridge is configured like other custom adapters. You can use the Jython or Jacl scripting language with the wsadmin scripting tool to define the user registry bridge in the federated repositories configuration.

Before you begin

Shut down WebSphere Application Server and the wsadmin command window.

Important: If you are migrating from the stand-alone user registry on the local operating system to federated repositories on the local operating system, you must first configure the current user registry under federated repositories. For more information, see Managing the realm in a federated repository configuration.

Authorization failures might occur if users or groups are mapped to roles before migration and you use those users or groups after migrating to user registry bridge. This situation occurs because the mapping contains registry-specific information. After migration, re-map the users or groups to avoid authorization failures.

About this task

For additional information about the commands to use for this topic, see IdMgrRepositoryConfig command group for the AdminTask object.

Use the following steps to add a user registry bridge to any federated repositories configuration and to any realm that is defined within the configuration.

Procedure

1. Start the wsadmin scripting tool. You can use the following command to start the wsadmin scripting tool:

```
wsadmin –conntype none
```

2. Use the createIdMgrCustomRepository command to add a new repository configuration for the user registry bridge.

   The following example configures a custom repository to use the com.ibm.ws.wim.adapter.urbridge.URBridge class and sets urbcustom as the identifier:

   Using Jython:

```
AdminTask.createIdMgrCustomRepository('-id urbcustom
-adapterClassName com.ibm.ws.wim.adapter.urbridge.URBridge')
```

   Using Jacl:

```
$AdminTask createIdMgrCustomRepository {-id urbcustom
-adapterClassName com.ibm.ws.wim.adapter.urbridge.URBridge}
```

   gotcha: The user registry bridge handles requests to one user registry only. Therefore, if you define multiple repositories, each user registry implementation must have a separate instance of the user registry bridge and you must define each implementation as a separate repository with a unique repository ID..

3. Optional: Add the necessary registry-specific properties as custom properties. Use the setIdMgrCustomProperty command repeatedly to add multiple properties. Use this command once per property to add multiple properties to your configuration. You must use both the name and value parameters to add the custom property for the specified repository. For example, to add a custom property of uniqueUserIdProperty, enter the following command:

   Using Jython:

```
AdminTask.setIdMgrCustomProperty('-id urbcustom
-name uniqueUserIdProperty -value "uniqueId"')
```

   Using Jacl:

```
$AdminTask setIdMgrCustomProperty {-id urbcustom
-name uniqueUserIdProperty -value "uniqueId"}
```

   To configure the user registry bridge to use a custom user registry, you must add the registryImplClass property and specify the exact registry implementation class. For example, specify com.xyz.abc.MyCustomRegistry as the value for the property.

   To configure the user registry bridge to use the local operating system user registry, do not specify the registryImplClass property. The user registry bridge identifies the underlying user registry implementation that is provided by WebSphere Application Server for the local operating system.

   You can set other optional properties as custom properties to define the mapping between federated repository properties and user registry properties, such as uniqueUserIdProperty, userSecurityNameProperty, userDisplayNameProperty, uniqueGroupIdProperty, groupSecurityNameProperty, and groupDisplayNameProperty. For more information about the available custom properties and their default values, see Security custom properties. To override any of these properties at the user registry level, configure the property as a custom property.

4. Add a base entry to the user registry bridge configuration. Use the addIdMgrRepositoryBaseEntry command to specify the name of the base entry for the specified repository. For example:

   Using Jython:

```
AdminTask.addIdMgrRepositoryBaseEntry('-id urbcustom
-name o=custom')
```

   Using Jacl:

```
$AdminTask addIdMgrRepositoryBaseEntry {-id urbcustom
-name o=custom}
```

5. Use the addIdMgrRealmBaseEntry command to add the base entry to the realm, which will link the realm with the repository.

   Note: The default realm name is defaultWIMFileBasedRealm. If this realm name was previously renamed, use the new realm name instead of defaultWIMFileBasedRealm. For example, to ensure consistency, you can set the realm name of the federated repository configuration to be the same name as the local operating system user registry as specified in the `security.xml` file. For information about how to set the realm name, see Realm configuration settings.

   Use the following command:

   Using Jython:

```
AdminTask.addIdMgrRealmBaseEntry('-name defaultWIMFileBasedRealm
-baseEntry o=custom')
```

   Using Jacl:

```
$AdminTask addIdMgrRealmBaseEntry {-name defaultWIMFileBasedRealm
-baseEntry o=custom}
```

6. Save your configuration changes. Enter the following commands to save the new configuration and close the wsadmin scripting tool:

   Using Jython:

```
AdminConfig.save()
exit
```

   Using Jacl:

```
$AdminConfig save
exit
```

7. Restart the application server.

Results

The following code is an example of a basic configuration in the `wimconfig.xml` file for a user registry bridge accessing a custom user registry:

```
<config:repositories adapterClassName="com.ibm.ws.wim.adapter.urbridge.URBridge" id="urbcustom">
<config:baseEntries name="o=custom"/>
  <config:CustomProperties name="registryImplClass" value="com.ibm.registry.impl.FileRegistrySample"/>
  <config:CustomProperties name="usersFile" value="${USER_PROPS}"/>
  <config:CustomProperties name="groupsFile" value="${GROUP_PROPS}"/>
</config:repositories>
```

In the previous example, the *${USER_PROPS}* and *${GROUP_PROPS}* variables are used to define the values of the custom properties.

You can use variables to define custom properties. However, these variables are resolved only in the WebSphere Application Server connected mode. For information about how to define environment variables, see Creating, editing, and deleting WebSphere variables.

User registry bridge for federated repositories:

The user registry bridge is a read-only adapter that provides an interface between federated repositories and an underlying user registry implementation, which can be either a local operating system user registry or a custom user registry implementation.

The user registry bridge enables IBM WebSphere Application Server applications to use your user registry implementation. It can work with any user registry that implements the com.ibm.websphere.security.UserRegistry interface, without knowing the details of its implementation. This capability makes the bridge versatile and allows it to connect to, and use, various registries.

The user registry bridge allows access to the same repository information without any platform-specific implementation. Thus, it eliminates the need to have a specialized user registry bridge for each operating system.

You can federate and configure the local operating system user registry, a custom user registry, or both, as a federated repository. The user registry bridge handles user registry-related requests from federated repositories, makes appropriate calls to the underlying user registry implementation, and returns data that is formatted according to the federated repositories specifications.

Therefore, to use the user registry bridge you must configure your user registry under federated repositories. This configuration can map the properties in the underlying user registry to the properties for the federated repository. You can also configure any user registry specific information, if required. For information about how to configure the user registry bridge, see Configuring the user registry bridge for federated repositories using wsadmin scripting.

The following figure illustrates the difference between configuring a federated repository user registry with and without the user registry bridge.

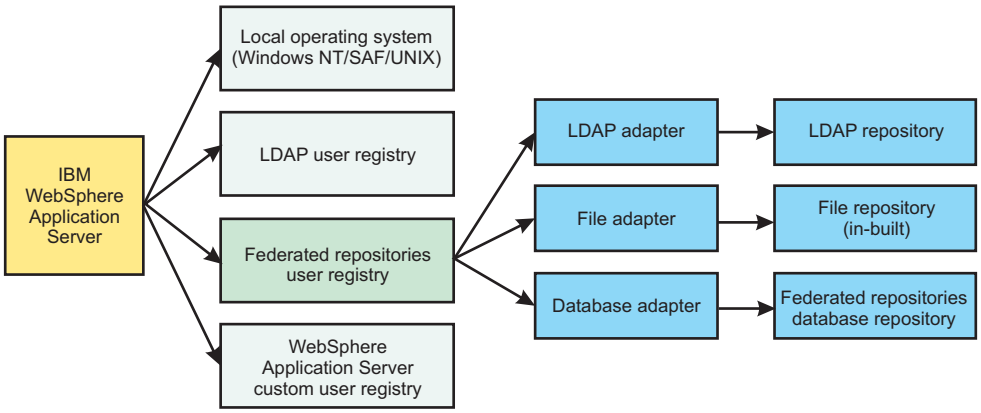


Figure 8. Configuring a federated repository user registry without the user registry bridge

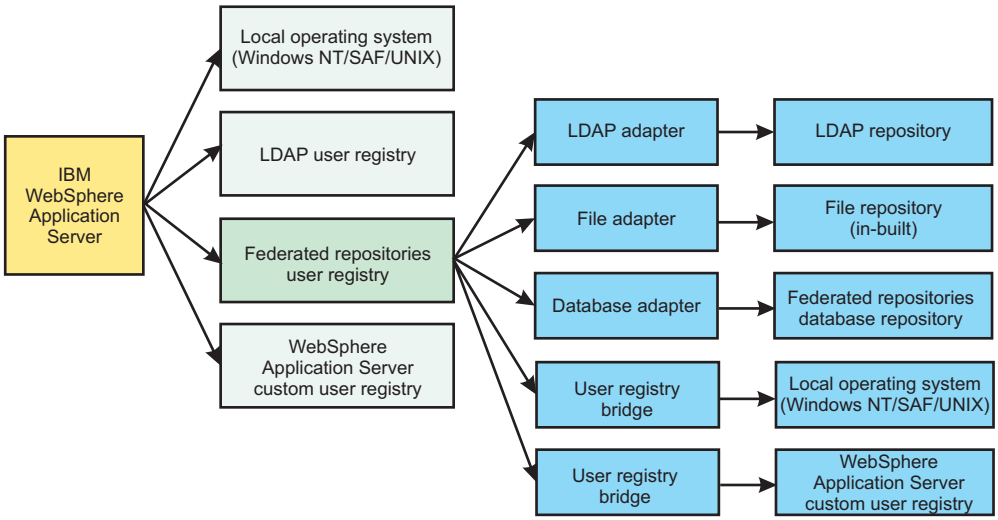


Figure 9. Configuring a federated repository user registry with the user registry bridge

As shown in the previous figure, using the same adapter, which is the user registry bridge, you can configure multiple user registries under federated repositories. For example, you can configure a local operating system user registry and one or more custom user registries.

Limitations

The following limitations exist:

- You can use the user registry bridge only for read-only operations, such as authentication and search functions. You cannot perform write operations such as create, delete, or modify users and groups. An attempt to perform write operations results in an exception, which notifies the user that the operation is not supported by the bridge. This limitation exists because the user registry bridge does not have direct access to the repository. Instead, the bridge uses an underlying existing user registry implementation that is read-only; hence, it might not be able to fulfill requests for certain properties that exist in the federated repositories.
- The user registry bridge does not support a stand-alone Lightweight Directory Access Protocol (LDAP) user registry. LDAP repositories are supported as a standard federated repositories adapter with read and write capabilities.
- Some of the properties that are placed in control data objects are not relevant to the user registry bridge as they are not applicable in the underlying repository.

– The properties ignored for GroupMembershipControl and GroupMemberControl data objects are searchBases, timeLimit, treeView, expression, and level.
– The properties ignored for SearchControl data objects are searchBases and timeLimit. The property part of the expression, such as uid and mail, is ignored as you can search WebSphere Application Server user registry entities with security names only. The expression is parsed to get the entity type and the pattern with which the search must be performed.

Supported user registries

WebSphere Application Server applications can access the user registry properties of the following user registry implementations as a read-only repository:

- Local operating system user registry
- Custom user registry

Configuring Lightweight Directory Access Protocol entity types in a federated repository configuration:

Follow this task to configure Lightweight Directory Access Protocol (LDAP) entity types in a federated repository configuration.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured. During LDAP configuration, based on the selected LDAP server type, some defaults and mappings are set in the configuration. When the selected LDAP server type is custom, no default is set, and you must set all of the mappings manually. To avoid setting all of the mappings manually, choose a non-custom LDAP server type (for example, IBM Directory Server or SunOne) which matches closely to your LDAP server.

   Note:

   - If you click **Add** to specify a new external repository, you must first complete the required fields and click **Apply** before you can proceed to the next step.
   - If you decide to use a custom LDAP server type, you must use the command-line interface to create the entity types. Read about IdMgrRepositoryConfig command group for the AdminTask object for more information.

     After you create the entity types, you can use the administrative console to modify these entities. You cannot use the administrative console to create entity types for a custom LDAP server type.
5. Under Additional properties, click **LDAP entity types**.
6. View the entity types that are supported by the member repositories, or select an entity type to view or change its configuration properties.
7. Supply the object classes that are mapped to this entity type in the Object classes field. LDAP entries that contain one or more of the object classes belong to this entity type.
8. Supply the search bases that are used to search this entity type. The search bases specified must be subtrees of the base entry in the repository. For example, you can specify the following search bases, where `o=ibm,c=us` is the base entry in the repository:

   `o=ibm,c=us` or `cn=users,o=ibm,c=us` or `ou=austin,o=ibm,c=us`

   In the preceding example, you cannot specify search bases `c=us` or `o=ibm,c=uk`.

   Delimit multiple search bases with a semicolon (;). For example:

`ou=austin,o=ibm,c=us;ou=raleigh,o=ibm,c=us`

9. Supply the LDAP search filter that is used to search this entity type.

   For example, use `(objectclass=ePerson)` to search for users or `(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)` to search for groups in an external LDAP repository.

   If a search filter is not specified, the object classes and the relative distinguished name (RDN) properties are used to generate the search filter. For information on RDN properties, see “Configuring supported entity types in a federated repository configuration” on page 1377.

Results

After completing these steps, LDAP entity types are configured for your LDAP repository.

What to do next

1. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
2. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
3. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Lightweight Directory Access Protocol entity types collection:

Use this page to list Lightweight Directory Access Protocol (LDAP) entity types that are supported by the member repositories or to select an LDAP entity type to view or change its configuration properties.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **LDAP entity types**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Entity type:

Specifies the entity type name.

Object classes:

Specifies the object classes that are mapped to this entity type. LDAP entries that contain one or more of the object classes belong to this entity type.

You cannot map multiple entity types to the same LDAP object class.

Lightweight Directory Access Protocol entity types settings:

Use this page to configure Lightweight Directory Access Protocol (LDAP) entity types that are supported by the member repositories.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **LDAP entity types**.
6. Select an entity type to view or change its configuration properties.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Entity type:

Specifies the entity type.

Object classes:

Specifies the object classes that are mapped to this entity type. LDAP entries that contain one or more of the object classes belong to this entity type.

You cannot map multiple entity types to the same LDAP object class.

Search bases:

Specifies the search bases that are used to search this entity type.

The search bases specified must be subtrees of the base entry in the repository. For example, you can specify the following search bases, where `o=ibm,c=us` is the base entry in the repository:

`o=ibm,c=us` or `cn=users,o=ibm,c=us` or `ou=austin,o=ibm,c=us`

In the preceding example, you cannot specify search bases `c=us` or `o=ibm,c=uk`.

Delimit multiple search bases with a semicolon (;). For example:

```
ou=austin,o=ibm,c=us;ou=raleigh,o=ibm,c=us
```

Search filter:

Specifies the LDAP search filter that is used to search this entity type.

For example, use `(objectclass=ePerson)` to search for users or `(&(cn=%v)(|(objectclass=groupOfNames)(objectclass=groupOfUniqueNames)))` to search for groups in an external LDAP repository.

If a search filter is not specified, the object classes and the relative distinguished name (RDN) properties are used to generate the search filter.

Configuring group attribute definition settings in a federated repository configuration:

Follow this task to configure group definition settings in a federated repository configuration.

Before you begin

Because group attribute definition settings apply only to a Lightweight Directory Access Protocol (LDAP) repository, you must first configure an LDAP repository. For more information, see “Managing repositories in a federated repository configuration” on page 1380.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.

   Note: If you click **Add** to specify a new external repository, you must first complete the required fields and click **Apply** before you can proceed to the next step.
5. Under Additional properties, click **Group attribute definition**.
6. Supply the name of the group membership attribute in the Name of group membership attribute field. Only one membership attribute can be defined for each LDAP repository.

   Every LDAP entry should have this attribute to indicate the groups to which this entry belongs. For example, memberOf is the name of the membership attribute that is used in Active Directory. The group membership attribute contains values that reference groups to which this entry belongs. If UserA belongs to GroupA, then the value of the memberOf attribute of UserA should contain the distinguished name of GroupA.

   If your LDAP server does not support the group membership attribute, then do not specify this attribute. The LDAP repository can look up groups by searching the group member attributes, though the performance might be slower.
7. Select the scope of the group membership attribute. The default value is Direct.

   Direct The membership attribute contains direct groups only. Direct groups are the groups that contain the member. For example, if Group1 contains Group2 and Group2 contains User1, then Group2 is a direct group of User1, but Group1 is not a direct group of User1.

   Nested
   The membership attribute contains both direct groups and nested groups.

   All The membership attribute contains direct groups, nested groups, and dynamic members.

Results

After completing these steps, group attribute definition settings are configured for your LDAP repository.

What to do next

1. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
2. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.

3. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Group attribute definition settings:

Use this page to specify the name of the group membership attribute. Every Lightweight Directory Access Protocol (LDAP) entry includes this attribute to indicate the group to which this entry belongs.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **Group attribute definition**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Name of group membership attribute:

Specifies the name of the group membership attribute. Only one membership attribute can be defined for each Lightweight Directory Access Protocol (LDAP) repository.

Every LDAP entry should have this attribute to indicate the groups to which this entry belongs. For example, memberOf is the name of the membership attribute that is used in Active Directory. The group membership attribute contains values that reference groups to which this entry belongs. If UserA belongs to GroupA, then the value of the memberOf attribute of UserA should contain the distinguished name of GroupA.

If your LDAP server does not support the group membership attribute, then do not specify this attribute. The LDAP repository can look up groups by searching the group member attributes, though the performance might be slower.

Scope of group membership attribute:

Specifies the scope of the group membership attribute.

Default: Direct

Range:

Direct The membership attribute contains direct groups only. Direct groups are the groups that contain the member. For example, if Group1 contains Group2 and Group2 contains User1, then Group2 is a direct group of User1, but Group1 is not a direct group of User1.

Nested The membership attribute contains both direct groups and nested groups.

All The membership attribute contains direct groups, nested groups, and dynamic members.

Configuring member attributes in a federated repository configuration:

Follow this task to configure member attributes in a federated repository configuration.

Before you begin

Because member attributes apply only to a Lightweight Directory Access Protocol (LDAP) repository, you must first configure an LDAP repository. For more information, see “Managing repositories in a federated repository configuration” on page 1380.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.

   Note: If you click **Add** to specify a new external repository, you must first complete the required fields and click **Apply** before you can proceed to the next step.
5. Under Additional properties, click **Group attribute definition**.
6. Under Additional properties, click **Member attributes**.
7. Click **New** to specify a new member attribute or **Delete** to remove a preconfigured member attribute.
8. Accept the default, or supply the name of the member attribute in the Name of member attribute field. For example, member and uniqueMember are two commonly used names of member attributes.

   The member attribute is used to store the values that reference members that the group contains. For example, a group type with an object class groupOfNames has a member attribute named member; group type with object class groupOfUniqueNames has a member attribute named uniqueMember. An LDAP repository supports multiple group types if multiple member attributes and their associated group object classes are specified.
9. Supply the object class of the group that uses this member attribute in the Object class field. If this field is not defined, this member attribute applies to all group object classes.
10. Select the scope of the member attribute. The default value is Direct.

    Direct The member attribute contains direct members only. Direct members are members that are directly contained by the group. For example, if Group1 contains Group2 and Group2 contains User1, then User1 is a direct member of Group2, but User1 is not a direct member of Group1.

    Nested
    The member attribute contains both direct members and nested members.

    All The member attribute contains direct members, nested members, and dynamic members.

Results

After completing these steps, member attributes are configured for your LDAP repository.

What to do next

1. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
2. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.

3. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Member attributes collection:

Use this page to list Lightweight Directory Access Protocol (LDAP) member attributes or to select a member attribute to view or change its configuration properties.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **Group attribute definition**.
6. Under Additional properties, click **Member attributes**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Name:

Specifies the name of the member attribute in LDAP. For example, member and uniqueMember are two commonly used names of member attributes.

The member attribute is used to store the values that reference members that the group contains. For example, a group type with an object class groupOfNames has a member attribute named member; group type with object class groupOfUniqueNames has a member attribute named uniqueMember. An LDAP repository supports multiple group types if multiple member attributes and their associated group object classes are specified.

Scope:

Specifies the scope of the member attribute.

Default: Direct

Range:

Direct The member attribute contains direct members only. Direct members are members that are directly contained by the group. For example, if Group1 contains Group2 and Group2 contains User1, then User1 is a direct member of Group2, but User1 is not a direct member of Group1.

Nested The member attribute contains both direct members and nested members.

All The member attribute contains direct members, nested members, and dynamic members.

Object class:

Specifies the object class of the group that uses this member attribute. If this field is not defined, this member attribute applies to all group object classes.

Member attributes settings:

Use this page to configure Lightweight Directory Access Protocol (LDAP) member attributes.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is pre-configured.
5. Under Additional properties, click **Group attribute definition**.
6. Under Additional properties, click **Member attributes**.
7. Click **New** to specify a new member attribute.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Name of member attribute:

Specifies the name of the member attribute in LDAP. For example, member and uniqueMember are two commonly used names of member attributes.

The member attribute is used to store the values that reference members that the group contains. For example, a group type with an object class groupOfNames has a member attribute named member; group type with object class groupOfUniqueNames has a member attribute named uniqueMember. An LDAP repository supports multiple group types if multiple member attributes and their associated group object classes are specified.

Object class:

Specifies the object class of the group that uses this member attribute. If this field is not defined, this member attribute applies to all group object classes.

Scope:

Specifies the scope of the member attribute.

Default: Direct

Range:

Direct The member attribute contains direct members only. Direct members are members that are directly contained by the group. For example, if Group1 contains Group2 and Group2 contains User1, then User1 is a direct member of Group2, but User1 is not a direct member of Group1.

Nested The member attribute contains both direct members and nested members.

All The member attribute contains direct members, nested members, and dynamic members.

Configuring dynamic member attributes in a federated repository configuration:

Follow this task to configure dynamic member attributes in a federated repository configuration.

Before you begin

Because dynamic member attributes apply only to a Lightweight Directory Access Protocol (LDAP) repository, you must first configure an LDAP repository. For more information, see “Managing repositories in a federated repository configuration” on page 1380.

About this task

A dynamic group defines its members differently than a static group. Instead of listing the members individually, the dynamic group defines its members using an LDAP search. The filter for the search is defined in a dynamic member attribute. For example, the dynamic group uses the structural objectclass groupOfURLs, or auxiliary objectclass ibm-dynamicGroup, and the attribute memberURL, to define the search using a simplified LDAP URL syntax:

```
ldap:///<base DN of search> ? ? <scope of search> ? <searchfilter>
```

The following is an example of the LDAP URL that defines all entries that are under `o=Acme` with the `objectclass=person`:

```
ldap:///o=Acme,c=US??sub?objectclass=person
```

If both member and dynamic member attributes are specified for the same group type, this group type is a hybrid group with both static and dynamic members.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select**Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.

   Note: If you click **Add** to specify a new external repository, you must first complete the required fields and click **Apply** before you can proceed to the next step.
5. Under **Additional Properties**, click **LDAP entity types**.
6. Click the link for **Group** entity type.
7. In the **Object Classes** field, add the entry for the object class, for example, groupOfUrls. Delimit multiple entries with a semicolon (;).
8. Click **OK**.
9. Under Additional properties, click **Group attribute definition**.
10. Under Additional properties, click **Dynamic member attributes**.
11. Click **New**to specify a new dynamic member attribute or **Delete** to remove a preconfigured dynamic member attribute.
12. Specify the name of the dynamic member attribute in the Name of dynamic member attribute field. The name of the dynamic member attribute defines the filter for dynamic group members in LDAP, for example, memberURL is the name of a commonly used dynamic member attribute.
13. Specify the object class of the group that contains the dynamic member attribute in the Dynamic object class field, for example, groupOfURLs. If this property is not defined, the dynamic member attribute applies to all group object classes.
14. Save your configuration changes in the administration console: Click **System administration > Save changes to master repository > Save**.
15. This next step involves using a wsadmin command and cannot be done through the administrative console. Start the wsadmin scripting tool and connect to a server, by using the following command:

```
wsadmin –username username –password password
```

16. Use the addIdMgrPropertyToEntityTypes command to add the dynamic member attribute specified in step 12 to the federated repositories schema. The dynamic member attribute needs to be added to the entity type Group in the federated repositories schema otherwise an error occurs while creating a group in an LDAP repository configured under federated repositories using the create() API and specifying the memberURL attribute and its value. The correctness of the value of the memberURL attribute is not validated because LDAP does not validate this.

    In the following example, the memberURL property is added to the entity type Group:

    ```
    $AdminTask addIdMgrPropertyToEntityTypes {-name memberURL -dataType String -entityTypeNames Group -repositoryIds re
    ```

17. Save your configuration changes.

    ```
    $AdminConfig save
    ```

Results

After completing these steps, dynamic member attributes are configured for your LDAP repository.

What to do next

1. After configuring the federated repositories, click **Security > Global security** to return to the Global security panel. Verify that Federated repositories is identified in the Current realm definition field. If Federated repositories is not identified, select **Federated repositories** from the Available realm definitions field and click **Set as current**. To verify the federated repositories configuration, click **Apply** on the Global security panel. If Federated repositories is not identified in the Current realm definition field, your federated repositories configuration is not used by WebSphere Application Server.
2. If you are enabling security, complete the remaining steps as specified in “Enabling security for the realm” on page 1186. As the final step, validate this setup by clicking **Apply** in the Global security panel.
3. Save, stop, and restart all the product servers (deployment managers, nodes, and Application Servers) for changes in this panel to take effect. If the server comes up without any problems, the setup is correct.

Dynamic member attributes collection:

Use this page to manage Lightweight Directory Access Protocol (LDAP) dynamic member attributes.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **Group attribute definition**.
6. Under Additional properties, click **Dynamic member attributes**.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Name:

Specifies the name of the attribute that defines the filter for dynamic group members in LDAP. For example, memberURL is the name of a commonly used dynamic member attribute.

If both member and dynamic member attributes are specified for the same group type, this group type is a hybrid group with both static and dynamic members.

A dynamic group defines its members differently than a static group. Instead of listing the members individually, the dynamic group defines its members using an LDAP search. The filter for the search is defined in a dynamic member attribute. For example, the dynamic group uses the structural objectclass groupOfURLs, or auxiliary objectclass ibm-dynamicGroup, and the attribute memberURL, to define the search using a simplified LDAP URL syntax:

```
ldap:///<base DN of search>??<scope of search>?<searchfilter>
```

The following is an example of the LDAP URL that defines all entries that are under `o=Acme` with the `objectclass=person`:

```
ldap:///o=Acme,c=US??sub?objectclass=person
```

Object class:

Specifies the object class of the group that contains this dynamic member attribute, for example, groupOfURLs. If this property is not defined, the dynamic member attribute applies to all group object classes.

Dynamic member attributes settings:

Use this page to configure Lightweight Directory Access Protocol (LDAP) dynamic member attributes.

To view this administrative console page, complete the following steps:

1. In the administrative console, click **Security > Global security**.
2. Under User account repository, select **Federated repositories** from the Available realm definitions field and click **Configure**.
3. Under Related items, click **Manage repositories**.
4. Click **Add** to specify a new external repository or select an external repository that is preconfigured.
5. Under Additional properties, click **Group attribute definition**.
6. Under Additional properties, click **Dynamic member attributes**.
7. Click **New** to specify a new dynamic member attribute.

When you finish adding or updating your federated repository configuration, go to the **Security > Global security** panel and click **Apply** to validate the changes.

Name of dynamic member attribute:

Specifies the name of the attribute that defines the filter for dynamic group members in LDAP. For example, memberURL is the name of a commonly used dynamic member attribute.

If both member and dynamic member attributes are specified for the same group type, this group type is a hybrid group with both static and dynamic members.

A dynamic group defines its members differently than a static group. Instead of listing the members individually, the dynamic group defines its members using an LDAP search. The filter for the search is defined in a dynamic member attribute. For example, the dynamic group uses the structural objectclass groupOfURLs, or auxiliary objectclass ibm-dynamicGroup, and the attribute memberURL, to define the search using a simplified LDAP URL syntax:

```
ldap:///<base DN of search>??<scope of search>?<searchfilter>
```

The following is an example of the LDAP URL that defines all entries that are under `o=Acme` with the `objectclass=person`:

```
ldap:///o=Acme,c=US??sub?objectclass=person
```

Dynamic object class:

Specifies the object class of the group that contains this dynamic member attribute, for example, groupOfURLs. If this property is not defined, the dynamic member attribute applies to all group object classes.

Standalone Lightweight Directory Access Protocol registries

A Standalone Lightweight Directory Access Protocol (LDAP) registry performs authentication using an LDAP binding.

WebSphere Application Server security provides and supports the implementation of most major LDAP directory servers, which can act as the repository for user and group information. These LDAP servers are called by the product processes for authenticating a user and other security-related tasks. For example, the servers are used to retrieve user or group information. This support is provided by using different user and group filters to obtain the user and group information. These filters have default values that you can modify to fit your needs. The custom LDAP feature enables you to use any other LDAP server, which is not in the product-supported list of LDAP servers, for its user registry by using the appropriate filters.

Note: The initial profile creation configures WebSphere Application Server to use a federated repositories security registry option with the file-based registry. This security registry configuration can be changed to use other options, including the stand-alone LDAP registry. Instead of changing from the federated repositories option to the stand-alone LDAP registry option under the User account repository configuration, consider employing the federated repositories option, which provides for LDAP configuration. Federated repositories provide a wide range of capabilities, including the ability to have one or multiple user registries. It supports federating one or more LDAPs in addition to file-based and custom registries. It also has improved failover capabilities, and a robust set of member (user and group) management capabilities. Federated repositories is required when you are using the new member management capabilities in WebSphere Portal 6.1 and above, and Process Server 6.1 and above. The use of federated repositories is required for following LDAP referrals, which is a common requirement in some LDAP server environments (such as Microsoft Active Directory).

It is recommended that you migrate from stand-alone LDAP registries to federated repositories. If you move to WebSphere Portal 6.1 and above, and or WebSphere Process Server 6.1 and above, you should migrate to federated repositories prior to these upgrades. For more information about federated repositories and its capabilities, read the Federated repositories topic. For more information about how to migrate to federated repositories, read the Migrating a stand-alone LDAP repository to a federated repositories LDAP repository configuration topic.

To use LDAP as the user registry, you need to know an administrative user name that is defined in the registry, the server host and port, the base distinguished name (DN) and, if necessary, the bind DN and the bind password. You can choose any valid user in the registry that is searchable and have administrative privileges. In some LDAP servers, the administrative users are not searchable and cannot be used, for example, `cn=root` in SecureWay. This user is referred to as WebSphere Application Server security server ID, server ID, or server user ID in the documentation. Being a server ID means a user has special privileges when calling some protected internal methods. Normally, this ID and password are used to log into the administrative console after security is turned on. You can use other users to log in if those users are part of the administrative roles.

When security is enabled in the product, the primary administrative user name and password are authenticated with the registry during the product startup. If authentication fails, the server does not start. It is important to choose an ID and password that do not expire or change often. If the product server user ID or password need to change in the registry, make sure that the changes are performed when all the product servers are up and running.

When the changes are done in the registry, use the steps that are described in “Configuring Lightweight Directory Access Protocol user registries” on page 1260. Change the ID, password, and other configuration information, save, stop, and restart all the servers so that the new ID or password is used by the product. If any problems occur starting the product when security is enabled, disable security before the server can start up. To avoid these problems, make sure that any changes in this panel are validated in the Global security panel. When the server is up, you can change the ID, password, and other configuration information and then enable security.

You can use the custom Lightweight Directory Access Protocol (LDAP) feature to support any LDAP server by setting up the correct configuration. However, support is not extended to these custom LDAP servers because many configuration possibilities exist.

The users and groups and security role mapping information is used by the configured authorization engine to perform access control decisions.

Dynamic groups and nested group support for LDAP:

Dynamic and nested groups simplify WebSphere Application Server security management and increase its effectiveness and flexibility.

Dynamic groups contain a group name and membership criteria:

- The group membership information is as current as the information on the user object.
- There is no need to manually maintain members on the group object.
- Dynamic groups are designed so an application does not need a large amount of information from the directory to find out if someone is a member of a group.

Nested groups enable the creation of hierarchical relationships that are used to define inherited group membership. A nested group is defined as a child group entry whose distinguished name (DN) is referenced by a parent group entry attribute.

You only need to assign a larger parent group if all nested groups share the same privilege. Assigning a role to a single parent group simplifies the run-time authorization table.

Dynamic groups and nested group support for the IBM Tivoli Directory Server

WebSphere Application Server supports all Lightweight Directory Access Protocol (LDAP) dynamic and nested groups when using IBM Tivoli Directory Server. This function is enabled by default by taking advantage of a new feature in IBM Tivoli Directory Server. IBM Tivoli Directory Server uses the ibm-allGroups forward-reference group attribute that automatically calculates all the group memberships including dynamic and recursive memberships for a user. Security directly locates a user group membership from a user object rather than indirectly search all the groups to match group members.

For more information, see “Configuring dynamic and nested group support for the IBM Tivoli Directory Server” on page 1282.

When you create groups, ensure that nested and dynamic group memberships work correctly.

Dynamic and nested group support for the SunONE or iPlanet Directory Server

The SunONE or iPlanet Directory Server uses two grouping mechanisms:

Groups
Entries that name other entries as a list of members or as a filter for members.

Roles Entries that name other entries as a list of members or as a filter for members. Additional functionality is provided by generating the nsrole attribute on each role member.

Three types of roles are available:

Filtered roles
Depends upon the attributes that are contained in each entry. Entries are members, if they match a specified Lightweight Directory Access Protocol (LDAP) filter. This role is equivalent to a dynamic group.

Nested roles
Creates roles that contain other roles. This role is equivalent to a nested group.

Managed roles
Explicitly assigns a role to member entries. This role is equivalent to a static group.

Refer to "Configuring dynamic and nested group support for the SunONE or iPlanet Directory Server" on page 1282 for more information.

Security failover among multiple LDAP servers:

WebSphere Application Server security can be configured to attempt failovers between multiple Lightweight Directory Access Protocol (LDAP) hosts.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

If the current active LDAP server is unavailable, WebSphere Application Server security attempts a failover to the first available LDAP host in the specified host list. The multiple LDAP servers can be replicas of the same master LDAP server, or they can be any LDAP host with the same schema, which contain data that is imported from the same LDAP Data Interchange Format (LDIF) file.

Whenever a failover occurs, WebSphere Application Server security always uses the first available LDAP server in the specified host list. For example, if there are four LDAP servers configured in the order of L1, L2, L3, and L4, L1 is treated as the primary LDAP server. The preference of connection is from L1 to L4. If, for example, WebSphere Application Server security is currently connected to L4, and failover or reconnection is necessary, WebSphere Application Server security first attempts to connect to L1, L2, and then L3 in that order until the connection is successful.

The current LDAP host name is logged in message CWSCJ0419I in the WebSphere Application Server log file, `SystemOut.log`. If you want to reconnect to the primary LDAP host, run the WebSphere Application Server MBean method, resetLDAPBindInfo, with `null,null` as the input.

To configure LDAP failover among multiple LDAP hosts, you must use wsadmin or ConfigService to include the backup LDAP host, which does not have a number limitation. The LDAP host that is displayed in the administrative console is the primary LDAP host, and is the first item listed in the LDAP host list in `security.xml`.

The WebSphere Application Server security realm name defaults to the primary LDAP host name that is displayed in the administrative console. It includes a trailing colon and a port number (if one exists). However, the custom property, com.ibm.websphere.security.ldap.logicRealm, can be added to override the default security realm name. Use the logicRealm name to configure each cell to have its own LDAP host for interoperability and backward compatibility, and to provide flexibility for adding or removing the LDAP host dynamically. If migrating from a previous installation, the new logicRealm name does not take effect until administrative security is enabled again. To be compatible with a previous release that does not

support logic realm, the logicRealm name has to be the same as that used by the previous installation (the LDAP host name, including a trailing colon and port number).

When LDAP failover is configured by associating a single hostname to multiple IP addresses through the use of a load balancer (which does that translation transparently to WebSphere Application Server), entering an invalid password can cause multiple LDAP bind retries. WebSphere Application Server retries and the load balancer routes requests to multiple replicas. With the default settings, the number of LDAP bind retries is equal to one more than the number of associated IP addresses. This means a single invalid login attempt can cause the LDAP account to be locked. If the com.ibm.websphere.security.ldap.retryBind custom property is set to `false`, LDAP bind calls are not retried.

The following Jacl example shows how to use wsadmin to add a backup LDAP host for failover:

```
#--------------------------------------------------------------------
# Main
#  This is a bi-modal script: it can be included in the wsadmin
#  command invocation like this:
#     wsadmin -f LDAPAdd.jacl ldaphost 800
#
#  or the script can be sourced from the wsadmin command line:
#     wsadmin> source LDAPAdd.jacl
#     wsadmin> LDAPAdd ldaphost 800
#
#  The script expects some parameters:
#      arg1 - LDAP Server host name
#      arg2 - LDAP Server port number
#--------------------------------------------------------------------
if { !($argc == 2)} {
   puts ""
   puts "LDAPAdd: This script requires 2 parameters: LDAP server host name and LDAP server port number"
   puts "For example: LDAPAdd ldaphost 389"
   puts ""
   return;
}
else {
   set ldapServer        [lindex $argv 0]
   set ldapPort          [lindex $argv 1]
   LDAPAdd $ldapServer $ldapPort
   return;
}
proc LDAPAdd {ldapServer ldapPort args} {
   global AdminConfig AdminControl ldapServer ldapPort
   set ldapServer  lindex $args 0
   set ldapPort  lindex $args 1
   global ldapUserRegistryId
   # Get the LDAP user registry object from the security configuration
   if { catch {$AdminConfig list LDAPUserRegistry} result } {
      puts stdout "\$AdminConfig list LDAPUserRegistry caught an exception $result\n"
      return
   }
   else {
      if {$result != {}} {
         set ldapUserRegistryId  lindex $result 0
      }
      else {
         puts stdout "\$AdminConfig list LDAPUserRegistry caught an exception $result\n"
         return;
      }
   }
   # Set the host and port values in Attrs2
   set Attrs2  list  list hosts  list  list  list host
   $ldapServer
   list port $ldapPort

   # Modify the LDAP configuration host object
   $AdminConfig modify $ldapUserRegistryId $Attrs2
   $AdminConfig save
}
```

The following Jython example shows how to use wsadmin to add a backup LDAP host for failover:

```
#--------------------------------------------------------------------
# Add ldap hostname and port
#     wsadmin -f LDAPAdd.py arg1 arg2
#
#  The script expects some parameters:
#      arg1 - LDAP Server hostname
#      arg2 - LDAP Server portnumber
#--------------------------------------------------------------------
import java

#-------------------------------------------------------
# get the line separator and use to do the parsing
```

```
# since the line separator on different platform are different
lineSeparator = java.lang.System.getProperty('line.separator')

#-------------------------------------------------------------------------------
# add LDAP host
#-------------------------------------------------------------------------------
def LDAPAdd (ldapServer, ldapPort):
    global AdminConfig, lineSeparator, ldapUserRegistryId
    try:
        ldapObject = AdminConfig.list("LDAPUserRegistry")
        if len(ldapObject) == 0:
            print "LDAPUserRegistry ConfigId was not found\n"
            return

        ldapUserRegistryId = ldapObject.split(lineSeparator)[0]
        print "Got LDAPUserRegistry ConfigId is " + ldapUserRegistryId + "\n"
    except:
        print "AdminConfig.list('LDAPUserRegistry') caught an exception\n"

    try:
        secMbeans = AdminControl.queryNames('WebSphere:type=SecurityAdmin,*')
        if len(secMbeans) == 0:
            print "Security Mbean was not found\n"
            return

        secMbean = secMbeans.split(lineSeparator)[0]
        print "Got Security Mbean is " + secMbean + "\n"
    except:
        print "AdminControl.queryNames('WebSphere:type=SecurityAdmin,*') caught an exception\n"


    attrs2 = [["hosts", [[["host", ldapServer], ["port", ldapPort]]]]]
    try:
        AdminConfig.modify(ldapUserRegistryId, attrs2)
        try:
            AdminConfig.save()
            print "Done setting up attributes values for LDAP User Registry"
            print "Updated was saved successfully\n"
        except:
            print "AdminConfig.save() caught an exception\n"
    except:
        print "AdminConfig.modify(" + ldapUserRegistryId + ", " + attrs2 + ") caught an exception\n"
    return

#-------------------------------------------------------------------------------
# Main entry point
#-------------------------------------------------------------------------------
if len(sys.argv) < 2 or len(sys.argv) > 3:
        print("LDAPAdd: this script requires 2 parameters: LDAP server hostname and LDAP server port number\n")
        print("e.g.: LDAPAdd ldaphost 389\n")
        sys.exit(1)
else:
        ldapServer = sys.argv[0]
        ldapPort = sys.argv[1]
        LDAPAdd(ldapServer, ldapPort)
        sys.exit(0)
```

Selecting an authentication mechanism

An *authentication mechanism* defines rules about security information, such as whether a credential is forwardable to another Java process, and the format of how security information is stored in both credentials and tokens. You can select and configure an authentication mechanism by using the administrative console.

About this task

Authentication is the process of establishing whether a client is who or what it claims to be in a particular context. A client can be either an end user, a machine, or an application. An authentication mechanism in WebSphere Application Server typically collaborates closely with a *user registry*. The user registry is the user and groups account repository that the authentication mechanism consults with when performing authentication. The authentication mechanism is responsible for creating a *credential*, which is an internal product representation of a successfully authenticated client user. Not all credentials are created equally. The abilities of the credential are determined by the configured authentication mechanism.

WebSphere Application Server provides three authentication mechanisms: Lightweight Third Party Authentication (LTPA), Kerberos, and RSA token authentication mechanism.

Security support for Kerberos as the authentication mechanism has been added for this release of WebSphere Application Server. Kerberos (KRB5) is a mature, flexible, open, and very secure network authentication protocol. Kerberos includes authentication, mutual authentication, message integrity and confidentiality and delegation features. KRB5 is used for Kerberos in the administrative console and in the **sas.client.props**, **soap.client.props** and **ipc.client.props** files.

The RSA token authentication mechanism is new to this release of WebSphere Application Server. It aids the flexible management objective to preserve the base profiles configurations and isolate them from a security perspective. This mechanism permits the base profiles managed by an administrative agent to have different Lightweight Third-Party Authentication (LTPA) keys, different user registries, and different administrative users.

Note: Simple WebSphere Authentication Mechanism (SWAM) is deprecated in this release. SWAM does not provide authenticated communication between different servers.

Authentication is required for enterprise bean clients and web clients when they access protected resources. Enterprise bean clients, like a servlet or other enterprise beans or a pure client, send the authentication information to a web application server using one of the following protocols:

- Common Secure Interoperability Version 2 (CSIv2)
- Secure Authentication Service (SAS)

Note: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Web clients use the HTTP or HTTPS protocol to send the authentication information.

The authentication information can be basic authentication (user ID and password), a credential token, or a client certificate. The web authentication is performed by the web authentication module.

You can configure web authentication for a web client by using the administrative console. Click **Security > Global security**. Under Authentication, expand **Web and SIP security** and click **General settings**. The following options exist for Web authentication:

Authenticate only when the URI is protected
Specifies that the web client can retrieve an authenticated identity only when it accesses a protected Uniform Resource Identifier (URI). WebSphere Application Server challenges the web client to provide authentication data when the web client accesses a URI that is protected by a J2EE role. This default option is also available in previous versions of WebSphere Application Server.

Use available authentication data when an unprotected URI is accessed
Specifies that the web client is authorized to call the `getRemoteUser`, `isUserInRole`, and `getUserPrincipal` methods; retrieves an authenticated identity from either a protected or an unprotected URI. Although the authentication data is not used when you access an unprotected URI, the authentication data is retained for future use. This option is available when you select the **Authentication only when the URI is protected** check box.

Authenticate when any URI is accessed
Specifies that the web client must provide authentication data regardless of whether the URI is protected.

Default to basic authentication when certificate authentication for the HTTPS client fails.
Specifies that WebSphere Application Server challenges the web client for a user ID and password when the required HTTPS client certificate authentication fails.

The enterprise bean authentication is performed by the Enterprise JavaBeans (EJB) authentication module.

The EJB authentication module resides in the CSIv2 and SAS layer.

The authentication module is implemented using the Java Authentication and Authorization Service (JAAS) login module. The web authenticator and the EJB authenticator pass the authentication data to the login module, which can use the following mechanisms to authenticate the data:

- Kerberos
- LTPA
- RSA token
- Simple WebSphere Authentication Mechanism (SWAM)

  Note: SWAM was deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release.

The authentication module uses the registry that is configured on the system to perform the authentication. Four types of registries are supported:

- Federated repositories
- Local operating system
- Standalone Lightweight Directory Access Protocol (LDAP) registry
- Stand-alone custom registry

External registry implementation following the registry interface that is specified by IBM can replace either the local operating system or the LDAP registry.

The login module creates a JAAS subject after authentication and stores the credential that is derived from the authentication data in the public credentials list of the subject. The credential is returned to the web authenticator or to the enterprise beans authenticator.

The web authenticator and the enterprise beans authenticator store the received credentials in the Object Request Broker (ORB) current for the authorization service to use in performing further access control checks. If the credentials are forwardable, they are sent to other application servers.

You can configure authentication mechanisms in the administrative console by doing the following:

Procedure

1. **Click Security > Global security.**
2. Under Authentication mechanisms and expiration, select an authentication mechanism to configure.

Lightweight Third Party Authentication

Lightweight Third Party Authentication (LTPA) is intended for distributed, multiple application server and machine environments. LTPA supports forwardable credentials and single sign-on (SSO). LTPA can support security in a distributed environment through cryptography. This support permits LTPA to encrypt, digitally sign, and securely transmit authentication-related data, and later decrypt and verify the signature.

Application servers can securely communicate using the LTPA protocol. It also provides the single sign-on (SSO) feature wherein a user is required to authenticate only once in a domain name system (DNS) domain and can access resources in other WebSphere Application Server cells without getting prompted. The realm names on each system in the DNS domain are case sensitive and must match identically.

For local OS, the realm name is the same as the host name.

For the Lightweight Directory Access Protocol (LDAP), the realm name is the `host:port` value of the LDAP server.

The LTPA protocol uses cryptographic keys to encrypt and decrypt user data that passes between the servers. These keys must be shared between the different cells for the resources in one cell to access resources in other cells, assuming that all the cells involved use the same LDAP or custom registry.

When using LTPA, a token is created with the user information and an expiration time and is signed by the keys. The LTPA token is time sensitive. All product servers that participate in a protection domain must have their time and date synchronized. If not, LTPA tokens appear prematurely expired and cause authentication or validation failures. Coordinated Universal Time (UTC) is used by default, and all other machines must have the same UTC time. Consult your operating system documentation for information regarding how to ensure this.

This token passes to other servers, in the same cell or in a different cell through cookies, for web resources when SSO is enabled, or through the authentication protocol layer for enterprise beans.

If the receiving servers share the same keys as the originating server, the token can be decrypted to obtain the user information, which then is validated to make sure that it has not expired and that the user information in the token is valid in its registry. On successful validation, the resources in the receiving servers are accessible after the authorization check.

Each server must have valid credentials. When the credentials expire, the server is required to communicate to the user registry to authenticate. User registry outages can cause server processes to hang, requiring them to be restarted to recover. Extending the time the LTPA token remains cached reduces this risk, but does present a slightly increased security risk to be considered when defining your security policies.

All of the WebSphere Application Server processes in a cell (deployment manager, nodes, application servers) share the same set of keys. If key sharing is required between different cells, export them from one cell and import them to the other. For security purposes, the exported keys are encrypted with a random generated key and a user-defined password is used to protect the keys. This same password is needed when importing the keys into another cell. The password is only used to protect the keys and is not used to generate the keys.

WebSphere Application Server supports the LTPA, Kerberos and the Simple WebSphere Authentication Mechanism (SWAM) protocols.

Note: SWAM is deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release.

When security is enabled during profile creation time, LTPA is configured by default.

LTPA requires that the configured user registry be a centrally shared repository such as LDAP or a Windows domain-type registry so that users and groups are the same, regardless of the machine.

The use of LTPA with the local OS user registry is only applicable to configurations where all of the servers reside on the same system.

Lightweight Third Party Authentication key sets and key set groups:

Key set groups contain lists of key sets and Lightweight Third Party Authentication (LTPA) key generation schedules. Each key set contains key references to keys in key stores.

Note: It is not recommended that you choose to generate new keys automatically . Keys should only be generated during off hours. Once keys are generated, you might need to export the keys and to import the keys to other WebSphere cells or IBM products in which the keys are required to be sync to communicate with each other.

The keys for some key configurations must be generated together. The LTPA key pair is referenced in one key set while the secret or private key is in a separate key set. When the key set group is created, the two key sets are added as members of the key set group. Key set group settings determine whether the keys for both key sets are generated together automatically or manually.

The key set group contains the following attributes:

- Member key sets
- Choice of either manual or automatic key generation in the member key sets
- Schedule for automatically generating keys

Configuring LTPA and working with keys

You must configure Lightweight Third Party Authentication (LTPA) when you set up security for the first time. LTPA is the default authentication mechanism for WebSphere Application Server. After you have configured LTPA you can generate LTPA keys manually or automatically.

Procedure

1. Configure LTPA and generate the first LTPA keys. Use the administrative console to configure LTPA or Kerberos when you set up security for the first time. The LTPA keys are generated automatically the first time. Read the Configuring the Lightweight Third Party Authentication mechanism article for more information.

   Application servers distributed in multiple nodes and cells can securely communicate using the LTPA protocol. Key set groups contain lists of key sets and LTPA authentication key generation schedules. Each key set contains key references to keys in key stores. To generate keys automatically, each key set must be a member of a key set group.

   Read the Lightweight Third Party Authentication key sets and key set groups article for more information.

   The keys for some key configurations must be generated together. The LTPA key pair is referenced in one key set while the secret or private key is in a separate key set. When the key set group is created, the two key sets are added as members of the key set group. Key set group settings determine whether the keys for both key sets are generated together automatically or manually.

   The key set group contains the following attributes:

   - Member key sets
   - Choice of either manual or automatic key generation in the member key sets
   - Schedule for automatically generating keys

2. Generate keys manually or automatically, and control the number of active keys. WebSphere Application Server generates Lightweight Third Party Authentication (LTPA) keys automatically during the first server startup. You can generate additional keys as you need them in the Authentication mechanisms and expiration panel.

   You can disable the automatic generation of new LTPA keys for key sets that are members of a key set group. Automatic generation creates new keys on a schedule that you specify when you configure a key set group, which manages one or more key sets. WebSphere Application Server uses key set groups to automatically generate cryptographic keys or multiple synchronized key sets.

   Generating keys manually or enabling or disabling the generation of keys are tasks that require you to recycle the node agents and application servers to accept the new keys. If any of the node agents are down, run a manual file synchronization utility from the node agent machine to synchronize the security configuration from the deployment manager.

   Key sets manage LTPA keys in a key store that is based on a key alias prefix. A key alias prefix is automatically generated when you generate a new key and store it in a key store. Key stores can contain multiple versions of keys for any given key alias prefix. You can specify a maximum number of active keys in the key set configuration.

   Read the Generating Lightweight Third Party Authentication keys article for more information.

3. Import and export keys. To support single sign-on (SSO) in WebSphere® Application Server across multiple WebSphere Application Server domains or cells, you must share the LTPA keys and the password among the domains. You can import LTPA keys from other domains and export keys to other domains.

   Note: You should disable automatic key generation if you import or export keys to or from another cell. This disabling causes the imported keys to get lost and the exported keys to no longer interoperate with this cell over time

   You must recycle the node agents and application servers to accept the new keys. If any of the node agents are down, run a manual file synchronization utility from the node agent machine to synchronize the security configuration from the deployment manager.

   Read the Importing Lightweight Third Party Authentication keys and Exporting Lightweight Third Party Authentication keys articles for more information.
4. Manage keys from multiple cells. You can specify the shared keys and configure the authentication mechanism that is used to exchange information between servers to import and export LTPA keys across multiple WebSphere® Application Server cells.

   You must start the server again for any changes you make to become active.

   Read the Managing LTPA keys from multiple WebSphere Application Server cells article for more information.

Kerberos (KRB5) authentication mechanism support for security

The Kerberos authentication mechanism enables interoperability with other applications (such as .NET, DB2 and others) that support Kerberos authentication. It provides single sign on (SSO) end-to-end interoperable solutions and preserves the original requester identity.

Note: Security support for Kerberos as the authentication mechanism was added for WebSphere Application Server Version 7.0. Kerberos is a mature, flexible, open, and very secure network authentication protocol. Kerberos includes authentication, mutual authentication, message integrity and confidentiality and delegation features. You can enable Kerberos on the server side. Support is provided to enable the rich Java client to use the Kerberos token for authentication to the WebSphere Application Server.

The following sections describe Kerberos authentication in more detail:

- “What is Kerberos?”
- “The benefits of having Kerberos as an authentication mechanism” on page 1417
- “Kerberos authentication in a single Kerberos realm environment” on page 1417
- “Kerberos authentication in a cross or trusted Kerberos realm environment” on page 1418
- “Things to consider before setting up Kerberos as the authentication mechanism for WebSphere Application Server” on page 1422
- “Support information for Kerberos authentication” on page 1423
- “Setting up Kerberos as the authentication mechanism for WebSphere Application Server” on page 1424
- “Setting up Kerberos as the authentication mechanism for the pure Java client” on page 1424

What is Kerberos?

Kerberos has withstood the test of time and is now at version 5.0. Kerberos enjoys wide spread platform support (for example, for Windows, Linux, Solaris, AIX, and z/OS) partly because the Kerberos source code is freely downloadable from the Massachusetts Institute of Technology (MIT) where it was originally created.

Kerberos is composed of three parts: a client, a server, and a trusted third party known as the Kerberos Key Distribution Center (KDC). The KDC provides authentication and ticket granting services.

The KDC maintains a database or repository of user accounts for all of the security principals in its realm. Many Kerberos distributions use file-based repositories for the Kerberos principal and policy DB and others use Lightweight Directory Access Protocol (LDAP) as the repository.

Kerberos does not support any notion of groups (that is, iKeys groups or groups of users or principals). The KDC maintains a long-term key for each principal in its accounts database. This long-term key is derived from the password of the principal. Only the KDC and the user that the principal represents should know what the long-term key or password is.

The benefits of having Kerberos as an authentication mechanism

The benefits of having Kerberos as the authentication mechanism for WebSphere Application Server include the following:

- The Kerberos protocol is a standard. This enables interoperability with other applications (such as .NET, DB2 and others) that support Kerberos authentication. It provides single sign on (SSO) end-to-end interoperable solutions and preserves the original requester identity.
- When using Kerberos authentication, the user clear text password never leaves the user machine. The user authenticates and obtains a Kerberos ticket granting ticket (TGT) from a KDC by using a one-way hash value of the user password. The user also obtains a Kerberos service ticket from the KDC by using the TGT. The Kerberos service ticket that represents the client identity is sent to WebSphere Application Server for authentication.
- A Java client can participate in Kerberos SSO using the Kerberos credential cache to authenticate to WebSphere Application Server.
- J2EE, web service, .NET and web browser clients that use the HTTP protocol can use the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) token to authenticate to the WebSphere Application Server and participate in SSO by using SPNEGO web authentication. Support for SPNEGO as the web authentication service is new to this release of WebSphere Application Server.

  Read about “Single sign-on for HTTP requests using SPNEGO web authentication” on page 1444 for more information.
- WebSphere Application Server can support both Kerberos and Lightweight Third-Party Authentication (LTPA) authentication mechanisms at the same time.
- Server-to-server communication using Kerberos authentication is provided.

Kerberos authentication in a single Kerberos realm environment

WebSphere Application Server supports Kerberos authentication in a single Kerberos realm environment as shown in the following figure:

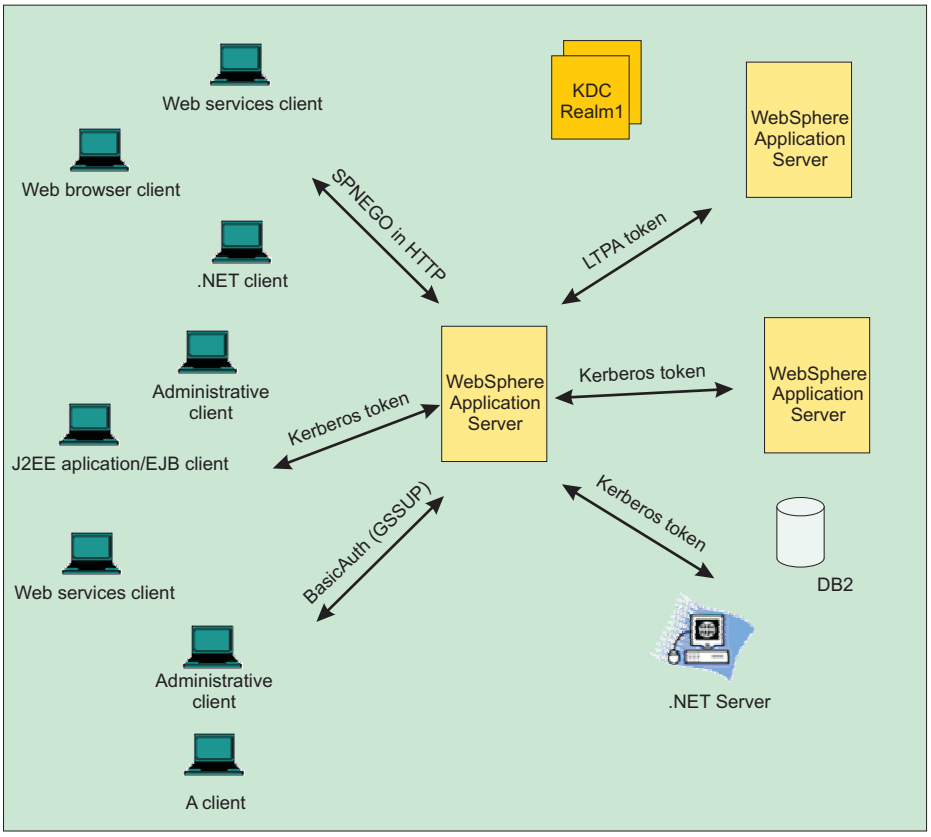


Figure 10. Kerberos authentication in a single Kerberos realm environment

When the WebSphere Application Server receives a Kerberos or SPNEGO token for authentication, it uses the Kerberos service principal (SPN) to establish a security context with a requestor. If a security context is established, the WebSphere Kerberos login module extracts a client GSS delegation credential, creates a Kerberos authentication token base on the Kerberos credential, and places them in the client subject with other tokens.

If the server must use a downstream server or back-end resources, it uses the client GSS delegation credential. If a downstream server does not support Kerberos authentication, the server uses the LTPA token instead of the Kerberos token. If a client does not include a GSS delegation credential in the request, the server uses the LTPA token for the downstream server . The Kerberos authentication token and principal are propagated to the downstream server as part of the security attributes propagation feature.

If the WebSphere Application Server and the KDC do not use the same user registry, then a JAAS custom login module might be required to map the Kerberos principal name to the WebSphere user name.

Kerberos authentication in a cross or trusted Kerberos realm environment

WebSphere Application Server also supports Kerberos authentication in a cross or trusted Kerberos realm environment as shown in the following figure:

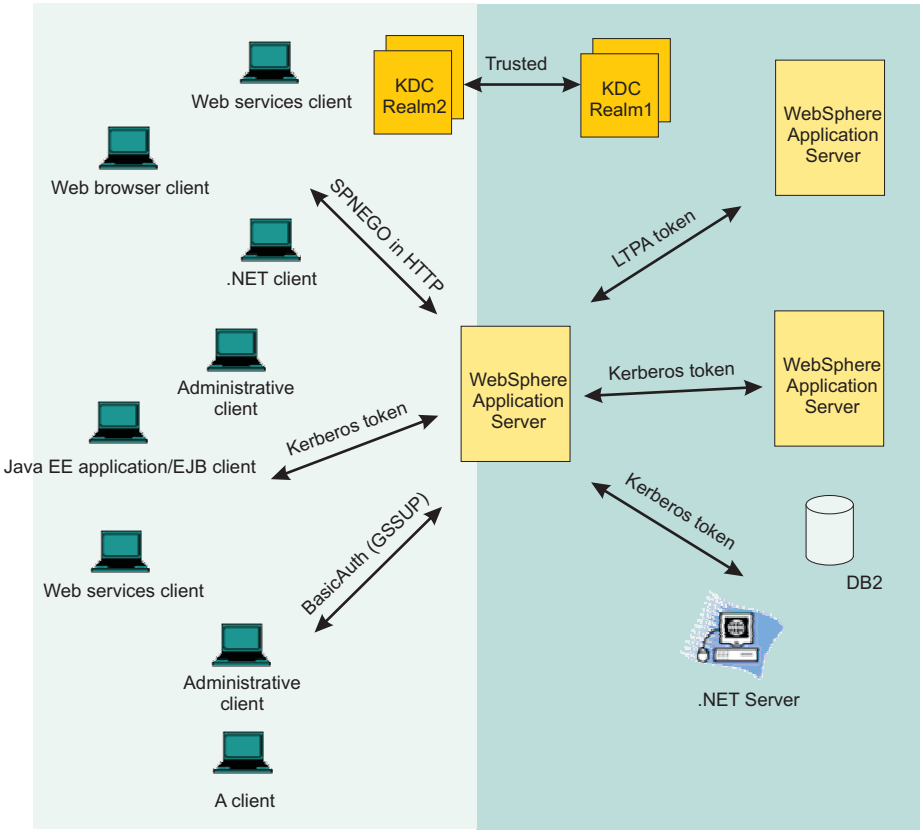


Figure 11. Kerberos authentication in a cross or trusted Kerberos realm environment

When the WebSphere Application Server receives a Kerberos or SPNEGO token for authentication, it uses the Kerberos service principal (SPN) to establish a security context with a requestor. If a security context is established, the WebSphere Kerberos login module always extracts a client GSS delegation credential and Kerberos ticket and places them in the client subject with other tokens.

If the server must use a downstream server or backend resources, it uses the client GSS delegation credential. If a downstream server does not support Kerberos authentication, the server uses the LTPA token instead of the Kerberos token. If a client does not include a GSS delegation credential in the request, the server uses the LTPA token for the downstream server . The Kerberos authentication token and principal are propagated to the downstream server as part of the security attributes propagation feature.

If the WebSphere Application Server and the KDC do not use the same user registry, then a JAAS custom login module might be required to map the Kerberos principal name to the WebSphere user name.

In this release of WebSphere Application Server, the new security multiple domains only support Kerberos at the cell level. All WebSphere Application Servers must be used by the same Kerberos realm. However, the clients and or backend resources (such as DB2, .NET server, and others) that support Kerberos authentication can have their own Kerberos realm. Only peer-to-peer and transitive trust cross-realm authentication are supported. The following steps must be performed for trusted Kerberos realms:

- The Kerberos trusted realm setup must be done on each of the Kerberos KDCs. See your Kerberos Administrator and User's guide for more information about how to set up a Kerberos trusted realm.
- The Kerberos configuration file might need to list the trusted realm.
- Add Kerberos trusted realms in the administrative console by clicking **Global security > CSIv2 outbound communications > Trusted authentication realms - outbound**.

The following figure shows a Java and administrative client that uses a Kerberos credential cache to authenticate to WebSphere Application Server with a Kerberos token in a trusted Kerberos realm:

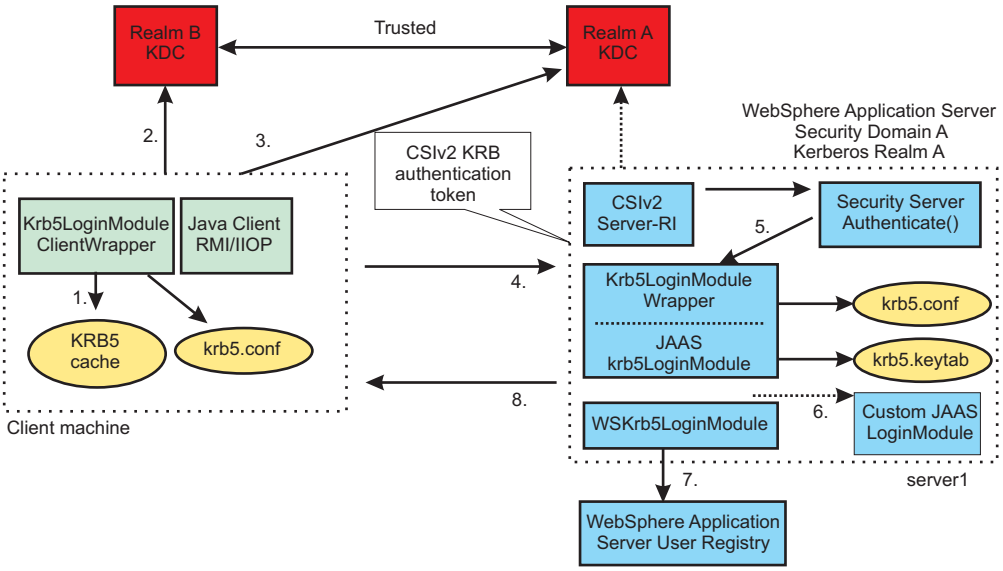


Figure 12. Using a Kerberos credential cache to authenticate to WebSphere Application Server with a Kerberos token in a trusted Kerberos realm

In the figure above, the following events occur:

1. The client uses the Kerberos credential cache if it exists.
2. The client requests a cross realm ticket (TGS_REQ) for `Realm A` from the `Realm B` KDC using the Kerberos credential cache.
3. The client uses a cross realm ticket to request Kerberos service ticket for `server1` (TGS_REQ) from the `Realm A` KDC.
4. The Kerberos token returned from the KDC (TGS_REP) is added to the CSIv2 message authentication token and sent to `server1` for authentication.
5. The server calls Krb5LoginModuleWrapper to establish security context with the client using the server Kerberos Service Principal Name (SPN) and keys from the **krb5.keytab** file. If the server successfully establishes a security context with the client, it always extracts the client GSS delegation credential and tickets and places them in the client subject.
6. Optionally, a custom JAAS Login Module might be needed if the KDC and WebSphere Application Server do not use the same user registry.
7. The user is validated with the user registry for WebSphere Application Server.
8. The results (success or failure) are returned to the client.

The following figure shows a Java and administrative client that uses a Kerberos principal name and password to authenticate to WebSphere Application Server with a Kerberos token:

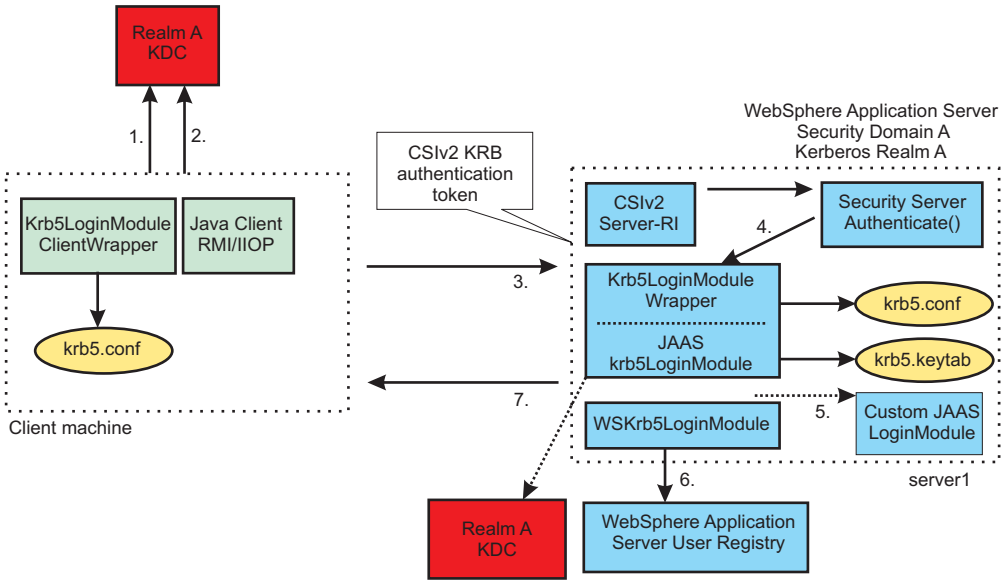


Figure 13. Using a Kerberos principal name and password to authenticate to WebSphere Application Server with a Kerberos token

In the figure above, the following events occur:

1. The client obtains the Kerberos granting ticket (TGT) from the KDC.
2. The client obtains a Kerberos service ticket for `server1` (TGS_REQ) using the TGT.
3. The Kerberos token returned from the KDC (TGS_REP) is added to the CSIv2 message authentication token and sent to `server1` for authentication.
4. The server calls Krb5LoginModuleWrapper to establish security context with the client using the server Kerberos Service Principal Name (SPN) and keys from the **krb5.keytab** file. If the server successfully establishes a security context with the client, it always extracts the client GSS delegation credential and tickets and places them in the client subject.
5. Optionally, a custom JAAS Login Module might be needed if the KDC and WebSphere Application Server do not use the same user registry.
6. The user is validated with the user registry for WebSphere Application Server.
7. The results are returned to the client.

The following figure shows server-to-server communications:

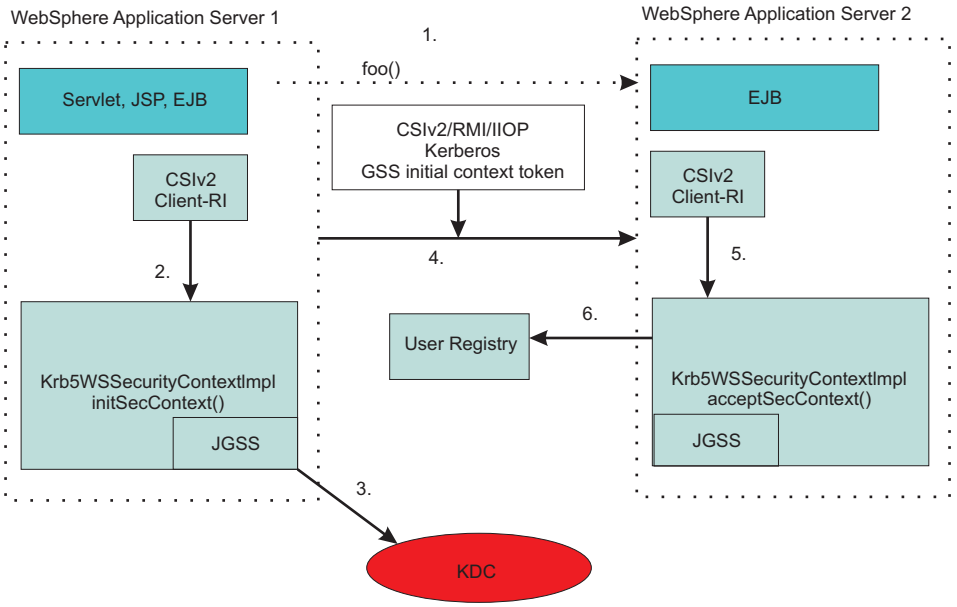


Figure 14. Server to server communications

When a WebSphere Application Server starts up, it uses the server ID and password to login to the KDC and then obtains the TGT. It then uses the TGT to request a service ticket to communicate with another server. If a WebSphere Application Server uses the internal server ID instead of the server ID and password, server-to-server communication is done using an LTPA token. In the figure above, the following events occur:

1. WebSphere Application Server 1 invokes a method, foo(), on an Enterprise JavaBeans (EJB) running in WebSphere Application Server 2.
2. `Server1` obtains a Kerberos service ticket for `Server2` (TGS_REQ) using the `Server1` TGT.
3. Same as step 2.
4. The Kerberos token returned from a KDC (TGS_REP) is added to the CSIv2 message authentication token and sent to `Server2` for authentication.
5. `Server2` calls the acceptSecContext() method to establish security context with `server1` using the `server2` Kerberos Service Principal Name (SPN) and keys from the **krb5.keytab** file. If `server2` successfully establishes a security context with `server1`, it always extracts the `server1` GSS delegation credential and tickets and places them in the subject.
6. The server id is validated with the WebSphere user registry.

Note: If a Java client application and the application server exist on the same machine and they use different Kerberos realm names, the run time uses the default realm name from the Kerberos configuration file. Alternatively, you can specify the realm name during the login process.

Things to consider before setting up Kerberos as the authentication mechanism for WebSphere Application Server

WebSphere Application Server now supports SPNEGO tokens in the HTTP header, Kerberos tokens, LTPA tokens and BasicAuth (GSSUP) for authentication.

To provide end-to-end Kerberos and end-to-end SPNEGO to Kerberos solutions, be aware of the following:

- The **Enabled delegation of Kerberos credentials** option must be selected. Read about Configuring Kerberos as the authentication mechanism using the administrative console for more information about this option.

- A client must obtain a ticket-granting ticket (TGT) with forwardable, address-less and renewable flags so that a target server can extract a client delegation Kerberos credential and use it for going to the downstream server.
- A client TGT that has an address can not be used for a downstream server, Data replication service (DRS) cache and cluster environments.
- See your Kerberos KDC platforms to make sure that it allows for client delegation Kerberos.
- For a long running application, a client should request a TGT with a renewable flag so that a target server can renew the delegation Kerberos.
- For a long-running application, ensure that the Kerberos ticket is valid for a period of time that is at least as long as the application runs. For example, if the application processes a transaction that takes 5 minutes, the Kerberos ticket must be valid for at least 5 minutes.
- Kerberos authentication and SPNEGO web authentication are both supported for Active Directory cross domain trusts within the same forest.
- In order for an administrative agent to use the Kerberos authentication mechanism, it must exchange an LTPA key with an administrative subsystem profile.
- If you plan to use the client delegation Kerberos credential for downstream authentication, make sure the client can request a service ticket that is greater than 10 minutes. If the client delegation Kerberos credential lifetime is less than 10 minutes, then the server attempts to renew it.

Note: The client, WebSphere Application Server and KDC machines must keep the clock synchronized. The best practice is to use a time server to keep all of the systems synchronized.

For this release of WebSphere Application Server, be aware of the following:

- Complete end-to-end Kerberos support with Tivoli Access Manager is available using the following KDCs:
  - z/OS
  - Microsoft (single or multi-realm)
  - AIX
  - Linux
- You can now configure and enable Kerberos cross realms for WebSphere Application Server and the thin client.
- WebSphere Application Server administrative function with Kerberos is limited by the following:
  - The preferred authentication mechanism for flexible management activities is the Rivest Shamir Adleman (RSA) authentication mechanism (by default).
  - Job Manager configured with Kerberos as the administrative authentication does not support Cross-Kerberos realms. They must be in the same Kerberos realm as registered nodes, or have the administrative authentication set to RSA
  - While Kerberos authentication is supported for administrative clients (wsadmin or Java clients) you should use the same KDC realm as the WebSphere Application Server it administers. Otherwise, a user id and password are recommended.
  - Mixed cell Kerberos and LTPA configuration is not supported when some of the nodes are WebSphere Application Server Release 6.x nodes or earlier.

Support information for Kerberos authentication

The following scenarios are supported:

- External domain trusts that are not on the same forests
- Domain trust within the same forest
- Kerberos realm trust

The following scenarios are not supported:

- Cross-forest trust
- Forest external trusts

Setting up Kerberos as the authentication mechanism for WebSphere Application Server

You must perform the steps in order as listed in “Setting up Kerberos as the authentication mechanism for WebSphere Application Server” to set up Kerberos as the authentication mechanism for WebSphere Application Server.

Note: Kerberos authentication mechanism on the server side must be done by the system administrator and on the Java client side by end users. The Kerberos keytab file must to be protected.

Setting up Kerberos as the authentication mechanism for the pure Java client

End users can optionally set up Kerberos authentication mechanism for the pure Java client. Read about Configuring a Java client for Kerberos authentication for more information.

Setting up Kerberos as the authentication mechanism for WebSphere Application Server

You must perform steps in this article in order to set up Kerberos as the authentication mechanism for WebSphere Application Server.

About this task

Note: Kerberos authentication mechanism on the server side must be done by the system administrator and on the Java client side by end users. The Kerberos keytab file must to be protected.

You must first ensure that the KDC is configured. See your Kerberos Administrator and User's guide for more information.

gotcha: When configuring the envar file for a z/OS KDC, order the encryption types from most secure to least secure for the SKDC_TKT_ENCTYPES environment variable. The z/OS KDC prefers to use the encryption types that are first in the list, from left to right.

You must perform the following steps in order to set up Kerberos as the authentication mechanism for WebSphere Application Server.

Procedure

1. Create a Kerberos service principal name and keytab file You can create a Kerberos service principal name and keytab file using Microsoft Windows, iSeries, Linux, Solaris, Massachusetts Institute of Technology (MIT) and z/OS operating systems key distribution centers (KDCs).

   Kerberos prefers servers and services to have a host-based service ID. The format of this ID is `<service name>/<fully qualified hostname>`. The default service name is `WAS`. For Kerberos authentication, the service name can be any strings that are allowed by the KDC. However, for SPNEGO web authentication, the service name must be HTTP. An example of a WebSphere Application Sever server ID is `WAS/myhost.austin.ibm.com`.

   Each host must have a server ID unique to the hostname. All processes on the same node share the same host-based service ID.

   A Kerberos administrator creates a Kerberos service principal name (SPN) for each node in the WebSphere cell. For example, for a cell with 3 nodes (such as `server1.austin.ibm.com`, `server2.austin.ibm.com` and `server3.austin.ibm.com`), the Kerberos administrator must create the following Kerberos service principals: `WAS/server1.austin.ibm.com`, `WAS/server2.austin.ibm.com` and `WAS/server3.austin.ibm.com`.

   The Kerberos keytab file (**krb5.keytab**) contains all of the SPNs for the node and must be protected. This file can be placed in the `config/cells/<cell_name>` directory

Read the Creating a Kerberos principal and keytab article for more information.

2. Create a Kerberos configuration file The IBM implementation of the Java Generic Security Service (JGSS) and KRB5 require a Kerberos configuration file (**krb5.conf** or **krb5.ini**) on each node or Java virtual machine (JVM). In this release of WebSphere Application Server, this configuration file should be placed in the `config/cells/<cell_name>` directory so that all application servers can access this file. If you do not have a Kerberos configuration file, use a **wsadmin** command to create one.

   Read the Creating a Kerberos configuration article for more information.

3. Configure Kerberos as the authentication mechanism for WebSphere Application Sever using the administrative console Use the administrative console to configure Kerberos as the authentication mechanism for the application server. When you have entered and applied the required information to the configuration, the Kerberos service principal name is formed as *<service name>/<fully qualified hostname>*`@KerberosRealm`, and is used to verify incoming Kerberos token requests.

   Read the Configuring Kerberos as the authentication mechanism using the administrative console article for more information.

4. Map a client Kerberos principal name to the WebSphere user registry ID You can map the Kerberos client principal name to the WebSphere user registry ID for both Simple and Protected GSS-API Negotiation (SPNEGO) web authentication and Kerberos authentication.

   Read the Mapping of a client Kerberos principal name to the WebSphere user registry ID article for more information.

5. Set up Kerberos as the authentication mechanism for the pure Java client (optional) A Java client can authenticate with WebSphere Application server with a Kerberos principal name and password or with the Kerberos credential cache (krb5Ccache).

   Read the Configuring a Java client for Kerberos authentication article for more information.

RSA token authentication mechanism

The Rivest Shamir Adleman (RSA) Authentication Mechanism is used to simplify the security environment for the Flexible Management Topology. It supports the ability to securely and easily register new servers to the Flexible Management topology. With the Flexible Management topology, you can submit and manage administrative jobs, locally or remotely, by using a job manager that manages applications, performs product maintenance, modifies configurations, and controls the application server runtime. The RSA authentication mechanism is only used for server-to-server administrative authentication, such as admin connector and file transfer requests. The RSA authentication mechanism does not replace LTPA or Kerboros for use by applications.

Note: The RSA token authentication mechanism aids the flexible management objective to preserve the base profiles configurations and isolate them from a security perspective. This mechanism permits the base profiles managed by an administrative agent to have different Lightweight Third-Party Authentication (LTPA) keys, different user registries, and different administrative users.

Important: The RSA token is not related to the RSA SecureId token. Please note that the application server does not provide support for SecureId.

Authentication is the process of establishing whether a client is who or what it claims to be in a particular context. A client can be either an end user, a machine, or an application. An authentication mechanism in WebSphere Application Server typically collaborates closely with a *user registry*. The user registry is the user and groups account repository that the authentication mechanism consults with when performing authentication. The authentication mechanism is responsible for creating a *credential*, which is an internal product representation of a successfully authenticated client user. Not all credentials are created equally. The abilities of the credential are determined by the configured authentication mechanism.

Authentication process

The RSA token authentication mechanism ensures that after the RSA root signer certificate (15 year lifetime) is exchanged between two administrative processes, there is no need to synchronize security

information among disparate profiles for administrative requests. The RSA personal certificate (1 year lifetime) is used to perform the cryptographic operations on the RSA tokens and can be verified by the long-lived RSA root. RSA token authentication is different from LTPA where keys are shared and if one side changes, all sides need to change. Since RSA token authentication is based on a PKI infrastructure, it benefits from the scalability and manageability of this technology in a large topology.

An RSA token has more advanced security features than LTPA; this includes a nonce value that makes it a one-time use token, a short expiration period (since it's a one-time use token), and trust, which is established based on certificates in the target RSA trust store.

RSA token authentication does not use the same certificates as used by Secure Sockets Layer (SSL). This is the reason RSA has it's own keystores. In order to isolate the trust established for RSA, the trust store, keystore, and root keystore, need to be different from the SSL configuration.

Note: SSL personal certificates given to pure clients are often signed by the same SSL root certificate used by servers, and this allows a pure client to send an RSA token to a server and act as an administrator. This should be avoided for the RSA token authentication mechanism. The RSA token authentication mechanism has its own root certificate which signs personal certificates that are used to encrypt and sign parts of the token.

The data stored in an RSA token is based on the identity of the client subject. The client subject can be based on LTPA or Kerberos, but the RSA token does not use this protection for administrative requests. The RSA token is easier to use while still maintaining a secure transportation of the identity. The data in an RSA token includes:

- Version
- Nonce
- Expiration
- Realm
- Principal
- Access ID
- Roles (not currently used)
- Groups
- Custom data

Custom data can be added to the WSCredential on the sending side (prior to going outbound) by creating a properties object, adding custom attributes, and adding this to the WSCredential in the following way.

```
import com.ibm.websphere.security.cred.WSCredential;

java.util.Properties props = new java.util.Properties();
props.setProperty("myAttribute", "myValue");
WSCredential.put ("customRSAProperties", props);
```

Once the Subject is created at the target process, you can get access to these attributes in the following way.

```
java.util.Properties props = (java.util.Properties) WSCredential.get("customRSAProperties");
```

This data is placed into a hash table at the target side and the hash table is used in a Java™ Authentication and Authorization Service (JAAS) login to obtain a subject at the target that contains the same attributes from the RSA token. With the target containing the same attributes from the RSA token, you can have a subject at the target side that is not from the same realm used by the target. For this authorization to succeed, a cross-realm mapping is required within the administrative authorization table unless the identity is a trusted server ID.

The figure (below) is an overview of the RSA token authentication mechanism and describes the process that takes place when a request is sent from a server-as-client to a target server. The server-as-client has an administrative subject on the thread that is used as input to create the RSA token. The other information needed is RSA public certificate of the target server. This certificate must be retrieved by making a “bootstrap” MBean request to the target process prior to sending any real requests. The target bootstrap request retrieves the public certificate from the target process. When creating an RSA token, the primary purpose of obtaining the target's public certificate is to encrypt the secret key. Only the target can decrypt the secret key, which is used to encrypt the user data.

The client's private key is used to sign both the secret key and the user data. The client's public key is embedded in the RSA token and validated at the target. If the client's public key is not trusted when calling the CertPath APIs at the target, the RSA token validation cannot continue. If the client's public key is trusted, it can be used to verify the secret key and user data signatures.

The basic goal is to convert the client subject into a subject at the target by securely propagating the required information. After the subject is generated at the target, the RSA authentication mechanism process is complete.

Configuring the RSA token authentication mechanism

You use the WebSphere Application Server administrative console to configure the Rivest Shamir Adleman (RSA) token authentication mechanism. The RSA token authentication mechanism can only be used for administrative requests. As such, the authentication mechanism choices for administrative authentication are part of the Global Security panel of the administrative console.

Before you begin

RSA token authentication mechanism is the default selection for the application server, administrative agent, and job manager profiles. LTPA is still the default for the deployment manager profile to preserve the same behavior for the existing topology.

About this task

You configure Lightweight Third-Party Authentication (LTPA) and Kerberos on the main authentication mechanism panels of the administrative console as well as configure RSA token authentication. During registration of a base profile with the administrative agent, the trusted certificates on both sides are updated with the root signer for the other. The same process occurs during registration of an administrative agent or deployment manager with a job manager. When removing the registration, the trusted signers are removed from both sides so that trust is no longer established.

By default, the RSA mechanism is set up correctly during the registration tasks, such as **registerNode** or **registerWithJobManager**. No further actions are necessary to establish trust within these environments. However, if you need to establish trust between two base servers or between two admin agents, for example, you can use the following steps to further configure the RSA token authentication mechanism:

Procedure

1. Click **Security > Global security** . Under Administrative security click the link to **Administrative authentication**.
2. Select the RSA token radio button. Select a data encryption keystore from the drop-down list. The option is recommend for flexible systems administration.
3. Optional: To exchange the root signers between two base servers:
   a. Select the root keystore from the Data encryption keystore drop-down list (such as NodeRSATokenRootStore).

b. Click **Extract Signer**.
c. Enter a fully-qualified name in the Certificate file name field.
d. Click **OK**.

4. Optional: Transfer the extracted root signer to the other server, and add it to that server's trusted signers keystore:
   a. Select the trusted keystore from the drop-down list (such as NodeRSATokenTrustedStore).
   b. Click **Add Signer**.
   c. Enter a unique name for the Alias.
   d. Enter a fully-qualified name for the signer key file.
   e. Click **OK**.
5. Enter the nonce cache timeout value.
6. Enter token timeout value.
7. Click **Apply** and **Save**.

Results

You have configured the RSA token authentication mechanism.

RSA token authentication settings:

Use this panel to configure RSA token authentication.

To view this administrative console page, click **Security > Global security**. Under Administrative security click **Administrative authentication**.

The administrative authentication method is used when an administrative process on this profile connects to another profile. If the primary authentication method is set to RSA token and that primary method fails, the system attempts to use the current application authentication method (which could be SWAM, Kerberos, or LTPA for example).

Note: SWAM is deprecated and will be removed in a future release.

RSA token (recommended for flexible systems administration):

RSA token is an authentication mechanism using certificates for signing and encryption portions of the security information being propagated.

Default: Enabled

Data encryption keystore:

This is the keystore that contains the personal certificate used to encrypt and sign RSA tokens.

Data type: text

Personal certificate for encryption:

This is the alias found in the Data encryption keystore that is used to encrypt and sign RSA tokens.

Data type: text

Trusted signers keystore:

This is the keystore used to contain signer certificates that can validate RSA tokens sent by other servers. The RSA token contains a sending certificate that needs to be validated by this trust store using a CertPath validation.

Data type: text

Nonce cache timeout:

Specifies the amount of time, in minutes, that the issued token is valid.

This field displays the maximum timeout, in minutes, for a token to be considered valid.

Data type: Integer
Default: 20
Minimum: 10
Maximum: Integer.MAX_VALUE

Token timeout:

Specifies the amount of time, in minutes, that the issued token is valid.

This field displays the maximum timeout, in minutes, for a token to be considered valid.

Data type: Integer
Default: 10
Minimum: 10
Maximum: Integer.MAX_VALUE

Only use the active application authentication mechanism (currently LTPA):

Select to encrypt authentication information so that the application server can send the data from one server to another in a secure manner.

The encryption of authentication information that is exchanged between servers involves the Lightweight Third-Party Authentication (LTPA) mechanism.

Kerberos:

Select to encrypt authentication information so that the application server can send the data from one server to another in a secure manner.

The encryption of authentication information that is exchanged between servers involves the Kerberos mechanism.

Note: Kerberos must be configured before this option can be selected.

RSA token certificate use:

The Rivest Shamir Adleman (RSA) token uses certificates in a similar way that Secure Sockets Layer (SSL) uses them. However, the trust established for SSL and RSA are different, and RSA certificates should not use SSL certificates and vice versa. The SSL certificates can be used by pure clients, and when used for the RSA mechanism would allow the client to send an RSA token to the server. The RSA token authentication mechanism is purely for server-to-server requests and should not be used by pure clients. The way to prevent this is to control the certificates used by RSA in such as a way so they are

never distributed to any clients. There is a different root certificate for RSA that prevents trust being established with clients who only need SSL certificates.

RSA root certificate

For each profile there is a root certificate stored in the **rsatoken-root-key.p12** keystore. The sole purpose of this RSA root certificate is to sign the RSA personal certificate which is stored in the **rsatoken-key.p12** keystore. The RSA root certificate has a default lifetime of 15 years. The signer from the RSA root certificate is shared with other processes to establish trust.

The keytool utility is available using the QShell Interpreter. Using the keytool utility, you can list the contents of these keystores and display the keyEntry (personal certificate). The example below illustrated how this is accomplished for the **rsatoken-root-key.p12** (RSA root certificate) and **rsatoken-key.p12** (RSA personal certificate).

```
${profile_root}\config\cells\${cellname}\nodes\${nodename}> keytool -list -v -keystore rsatoken-root-key.p12
–storepass WebAS -storetype PKCS12

Alias name: root
Entry type: keyEntry
Certificate[1]:
Owner: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60AACell04, OU=BIRKT60AANode04, O=IBM, C=US
Issuer: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60AACell04, OU=BIRKT60AANode04, O=IBM, C=US
Serial number: 3474fccaf789d
Valid from: 11/12/07 2:50 PM until: 11/7/27 2:50 PM
Certificate fingerprints:
         MD5:  7E:E6:C7:E8:40:4E:9B:96:5A:66:E5:0B:37:0B:08:FD
         SHA1: 36:94:81:55:C4:48:83:27:89:C7:16:D2:AD:3D:3E:67:DF:1D:6E:87

${profile_root}\config\cells\${cellname}\nodes\${nodename}> keytool -list -v -keystore rsatoken-key.p12
–storepass WebAS -storetype PKCS12

Alias name: default
Entry type: keyEntry
Certificate[1]:
Owner: CN=9.41.62.64, OU=BIRKT60AACell04, OU=BIRKT60AANode04, O=IBM, C=US
Issuer: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60AACell04, OU=BIRKT60AANode04, O=IBM, C=US
Serial number: 3475073488921
Valid from: 11/12/07 2:50 PM until: 11/11/08 2:50 PM
Certificate fingerprints:
         MD5:  FF:1C:42:E3:DA:FF:DC:A4:35:B2:33:30:D1:6E:E0:19
         SHA1: A4:FB:9D:7B:A1:5B:6A:37:9F:20:BD:B2:BD:98:FA:68:71:57:28:62
Certificate[2]:
Owner: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60AACell04, OU=BIRKT60AANode04, O=IBM, C=US
Issuer: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60AACell04, OU=BIRKT60AANode
04, O=IBM, C=US
Serial number: 3474fccaf789d
Valid from: 11/12/07 2:50 PM until: 11/7/27 2:50 PM
Certificate fingerprints:
         MD5:  7E:E6:C7:E8:40:4E:9B:96:5A:66:E5:0B:37:0B:08:FD
         SHA1: 36:94:81:55:C4:48:83:27:89:C7:16:D2:AD:3D:3E:67:DF:1D:6E:87
```

The purpose of the RSA personal certificate is to sign and encrypt information in the RSA token. The RSA personal certificate has a default lifetime of one year because it is used to sign and encrypt data that is transmitted over the wire. Refreshing the certificate is performed by the certificate expiration monitor, which is used for any other certificate in the system including SSL certificates.

RSA token trust is established when the **rsatoken-trust.p12** of the target process contains the signer of the root certificate of the client process that sends a token. Inside the RSA token is the public certificate of the client, which must be validated at the target before being used to decrypt data. The validation of the client's public certificate is performed using the CertPath APIs, which use the **rsatoken-trust.p12** as the source of certificates used during the validation.

The following example shows the use of the keytool utility to list the **rsatoken-trust.p12** keystore.

Note: This trust store contains three trustedCertEntry (public certificate) entries. The root public certificate from the administrative agent, a root public certificate from a job manager to which it is registered, and a root public certificate from a base profile to which it is registered.

```
${profile_root}\config\cells\${cellname}\nodes\${nodename}> keytool -list -v -keystore rsatoken-trust.p12
–storepass WebAS -storetype PKCS12

Alias name: root
Entry type: trustedCertEntry

Owner: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60AACell04, OU=BIRKT60AANode04, O=IBM, C=US
Issuer: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60AACell04, OU=BIRKT60AANode04, O=IBM, C=US
Serial number: 3474fccaf789d
Valid from: 11/12/07 2:50 PM until: 11/7/27 2:50 PM
Certificate fingerprints:
         MD5:  7E:E6:C7:E8:40:4E:9B:96:5A:66:E5:0B:37:0B:08:FD
         SHA1: 36:94:81:55:C4:48:83:27:89:C7:16:D2:AD:3D:3E:67:DF:1D:6E:87
*******************************************
Alias name: cn=9.41.62.64, ou=root certificate, ou=birkt60jobmgrcell02, ou=birkt
60jobmgr02, o=ibm, c=us
Entry type: trustedCertEntry

Owner: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60JobMgrCell02, OU=BIRKT60JobMgr02, O=IBM, C=US
Issuer: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60JobMgrCell02, OU=BIRKT60JobMgr02, O=IBM, C=US
Serial number: 34cc4c5d71740
Valid from: 11/12/07 4:30 PM until: 11/7/27 4:30 PM
Certificate fingerprints:
         MD5:  AB:65:3A:04:5B:C7:6D:A8:B1:98:B9:7B:65:A8:FA:F8
         SHA1: C0:83:FE:D0:B6:30:FB:A1:10:41:4B:8E:50:4B:78:40:0F:E5:E3:35
*******************************************
Alias name: birkt60node19_signer
Entry type: trustedCertEntry

Owner: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60Node15Cell, OU=BIRKT60Node19, O=IBM, C=US
Issuer: CN=9.41.62.64, OU=Root Certificate, OU=BIRKT60Node15Cell, OU=BIRKT60Node19, O=IBM, C=US
Serial number: 34825d997fda3
Valid from: 11/12/07 3:06 PM until: 11/7/27 3:06 PM
Certificate fingerprints:
         MD5:  66:61:CE:7C:C7:44:8B:A7:23:FF:1B:68:E4:AC:24:55
         SHA1: 25:E0:6B:D9:60:BB:67:5B:C6:67:BD:02:2C:54:E3:DA:24:E5:31:A3
*******************************************
```

You can use the WebSphere Application Server certificate management tools to create a new personal certificate, and then replace the RSA personal certificate in the rsa-key.p12 and the public key in the rsa-trust.p12 with this newly created personal certificate. If you replace the RSA personal certificate prior to federation to an administrative agent or job manager, the exchange of certificates is done for you. If you change the certificate after federation, you need to make sure the rsa-trust.p12 on the administrative agent or job manager is updated with the signer for your new certificate to establish trust.

Simple WebSphere authentication mechanism (deprecated)

The Simple WebSphere authentication mechanism (SWAM) defines rules about security information and the format of how security information is stored in both credentials and tokens. SWAM is intended for simple, non-distributed, single application server runtime environments.

Note: SWAM was deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release.

The single application server restriction is due to the fact that SWAM does not support *forwardable* credentials. If a servlet or enterprise bean in application server process 1, invokes a remote method on an enterprise bean living in another application server process 2, the identity of the caller identity in process 1

is not transmitted to server process 2. What is transmitted is an unauthenticated credential, which, depending on the security permissions configured on the EJB methods, can cause authorization failures.

Because SWAM is intended for a single application server process, single sign-on (SSO) is not supported.

The SWAM authentication mechanism is suitable for simple environments, software development environments, or other environments that do not require a distributed security solution.

Message layer authentication

Defines the credential information and sends that information across the network so that a receiving server can interpret it.

When you send authentication information across the network using a token the transmission is considered message layer authentication because the data is sent with the message inside a service context.

A pure Java client uses Kerberos (KRB5) or basic authentication, or Generic Security Services Username Password (GSSUP), as the authentication mechanism to establish client identity.

However, a servlet can use either basic authentication (GSSUP) or the authentication mechanism of the server, Kerberos (KRB5) or Lightweight Third Party Authentication (LTPA), to send security information in the message layer. Use KRB5 or LTPA by authenticating or by mapping the basic authentication credentials to the security mechanism of the server.

The security token that is contained in a token-based credential is authentication mechanism-specific. The way that the token is interpreted is only known by the authentication mechanism. Therefore, each authentication mechanism has an object ID (OID) representing it. The OID and the client token are sent to the server, so that the server knows which mechanism to use when reading and validating the token. The following list contains the OIDs for each mechanism:

BasicAuth (GSSUP): oid:2.23.130.1.1.1
KRB5: OID: 1.2.840.113554.1.2.2
LTPA: oid:1.3.18.0.2.30.2
SWAM: No OID because it is not forwardable

Note: SWAM is deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release.

On the server, the authentication mechanisms can interpret the token and create a credential, or they can authenticate basic authentication data from the client, and create a credential. Either way, the created credential is the *received* credential that the authorization check uses to determine if the user has access to invoke the method. You can specify the authentication mechanism by using the following property on the client side:

- com.ibm.CORBA.authenticationTarget

Basic authentication (BasicAuth) and KRB5 are currently the only valid values. You can configure the server through the administrative console.

Note: When **perform basic authentication** is enabled, if the client is not similarly configured (and does not pass a credential such as a user ID and password).

Configuring authentication retries

Situations occur where you want a prompt to display again if you entered your user ID and password incorrectly or you want a method to retry when a particular error occurs back at the client. If you can

correct the error by information at the client side, the system automatically performs a retry without the client seeing the failure, if the system is configured appropriately.

Some of these errors include:
- Entering a user ID and password that are not valid
- Having an expired credential on the server
- Failing to find the stateful session on the server

By default, authentication retries are enabled and perform three retries before returning the error to the client. Use the com.ibm.CORBA.authenticationRetryEnabled property (`True` or `False`) to enable or disable authentication retries. Use the com.ibm.CORBA.authenticationRetryCount property to specify the number of retry attempts.

Integrating third-party HTTP reverse proxy servers

These steps are required to use a trust association interceptor with a reverse proxy security server.

About this task

WebSphere Application Server enables you to use multiple trust association interceptors. The application server uses the first interceptor that can handle the request.

Procedure

1. Access the administrative console.

   Type `http://`*`server_name`*`:`*`port_number`*`/ibm/console` in a web browser.

   Port 9060 is the default port number for accessing the administrative console. During installation, however, you might have specified a different port number. Use the appropriate port number.
2. Click **Security > Global security**.
3. Under Web and SIP security, click **Trust association**.
4. Select the **Enable trust association** option.
5. Under Additional properties, click **Interceptors**. The default value appears.
6. Verify that the appropriate trust association interceptors are listed.

Results

Trust association is enabled.

What to do next

1. If you are enabling security, make sure that you complete the remaining steps for enabling security.
2. Save, stop and restart all of the product servers (deployment managers, nodes and application servers) for the changes to take effect.

Trust associations

Trust association enables the integration of IBM WebSphere Application Server security and third-party security servers. More specifically, a reverse proxy server can act as a front-end authentication server while the product applies its own authorization policy onto the resulting credentials that are passed by the proxy server.

Demand for such an integrated configuration has become more compelling, especially when a single product cannot meet all of the customer needs or when migration is not a viable solution. This article provides a conceptual background behind the approach.

In this setup, WebSphere Application Server is used as a back-end server to further exploit its fine-grained access control. The reverse proxy server passes the HTTP request to WebSphere Application Server that includes the credentials of the authenticated user. WebSphere Application Server then uses these credentials to authorize the request.

Trust association model

The idea that WebSphere Application Server can support trust association implies that the product application security recognizes and processes HTTP requests that are received from a reverse proxy server. WebSphere Application Server and the proxy server engage in a contract in which the product gives its full trust to the proxy server and the proxy server applies its authentication policies on every web request that is dispatched to WebSphere Application Server. This trust is validated by the interceptors that reside in the product environment for every request received. The method of validation is agreed upon by the proxy server and the interceptor.

Running in trust association mode does not prohibit WebSphere Application Server from accepting requests that did not pass through the proxy server. In this case, no interceptor is needed for validating trust.

WebSphere Application Server supports the following trust association interceptor (TAI) interfaces:

com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus
: This TAI interceptor implementation that implements the new WebSphere Application Server interface supports WebSphere Application Server Version 5.1.1 and later. The interface supports WebSEAL Version 5.1, but does not support WebSEAL Version 4.1. For an explanation of security attribute propagation, see “Security attribute propagation” on page 1544.

com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl
: This interceptor is new to this release. SPNEGO has replaced SPNEGO TAI as the web authenticator for WebSphere Application Server.

IBM WebSphere Application Server: WebSEAL Integration

The integration of WebSEAL and WebSphere Application Server security is achieved by placing the WebSEAL server at the front-end as a reverse proxy server. From a WebSEAL management perspective, a junction is created with WebSEAL on one end, and the product web server on the other end. A junction is a logical connection that is created to establish a path from the WebSEAL server to another server.

In this setup, a request for web resources that are stored in a protected domain of the product is submitted to the WebSEAL server where it is authenticated against the WebSEAL security realm. If the requesting user has access to the junction, the request is transmitted to the WebSphere Application Server HTTP server through the junction, and then to the application server.

Meanwhile, WebSphere Application Server validates every request that comes through the junction to ensure that the source is a trusted party. This process is referenced as *validating the trust* and it is performed by a WebSEAL product-designated interceptor. If the validation is successful, WebSphere Application Server authorizes the request by checking whether the client user has the required permissions to access the web resource. If so, the web resource is delivered to the WebSEAL server through the web server, which then gives the resource to the client user.

WebSEAL server

The policy director delegates all of the web requests to its web component, the WebSEAL server. One of the major functions of the server is to perform authentication of the requesting user. The WebSEAL server

consults a Lightweight Directory Access Protocol (LDAP) directory. It can also map the original user ID to another user ID, such as when global single sign-on (GSO) is used.

For successful authentication, the server plays the role of a client to WebSphere Application Server when channeling the request. The server needs its own user ID and password to identify itself to WebSphere Application Server. This identity must be valid in the security realm of WebSphere Application Server. The WebSEAL server replaces the basic authentication information in the HTTP request with its own user ID and password. In addition, WebSphere Application Server must determine the credentials of the requesting client so that the application server has an identity to use as a basis for its authorization decisions. This information is transmitted through the HTTP request by creating a header called `iv-creds`, with the Tivoli Access Manager user credentials as its value.

HTTP server

The junction that is created in the WebSEAL server must get to the HTTP server that serves as the product front end. However, the HTTP server is shielded from knowing that trust association is used. As far as it is concerned, the WebSEAL product is just another HTTP client, and as part of its normal routines, it sends the HTTP request to the product. The only requirement on the HTTP server is a Secure Sockets Layer (SSL) configuration using server authentication only. This requirement protects the requests that flow within the junction.

Web collaborator

When trust association is enabled, the web collaborator manages the interceptors that are configured in the system. The web collaborator loads and initializes these interceptors when you restart your servers. When a request is passed to WebSphere Application Server by the Web server, the web collaborator eventually receives the request for a security check. Two actions must take place:

1. The request must be authenticated.
2. The request must be authorized.

The web authenticator is called to authenticate the request by passing the HTTP request. If successful, a good credential record is returned by the authenticator, which the web collaborator uses to base its authorization for the requested resource. If the authorization succeeds, the web collaborator indicates to WebSphere Application Server that the security check has succeeded and that the requested resource can be served.

Web authenticator

The web authenticator is asked by the web collaborator to authenticate a given HTTP request. Knowing that trust association is enabled, the task of the web authenticator is to find the appropriate trust association interceptor to direct the request for processing. The web authenticator queries every available interceptor. If no target interceptor is found, the web authenticator processes the request as though trust association is not enabled.

Note:

> WebSphere Application Server Version 4 through WebSphere Application Server Version 6.*x* support the com.ibm.websphere.security.TrustAssociationInterceptor.java interface. WebSphere Application Server Version 7.0.*x* and later supports the com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl interface.

Trust association interceptor interface

The intent of the trust association interceptor interface is to have reverse proxy security servers (RPSS) exist as the exposed entry points to perform authentication and coarse-grained authorization, while WebSphere Application Server enforces further fine-grained access control. Trust associations improve security by reducing the scope and risk of exposure.

In a typical e-business infrastructure, the distributed environment of a company consists of web application servers, web servers, existing systems, and one or more RPSS, such as the Tivoli WebSEAL product. Such reverse proxy servers, front-end security servers, or security plug-ins registered within web servers, guard the HTTP access requests to the web servers and the web application servers. While protecting access to the Uniform Resource Identifiers (URIs), these RPSS perform authentication, coarse-grained authorization, and request routing to the target application server.

When a web server, such as an IBM HTTP Server, uses a TAI to communicate with WebSphere Application Server, sometimes it is essential for the TAI to know whether a request came through a web server or came directly to WebSphere Application Sever. Therefore the WebSphere Application Server Web container uses three HttpServletRequest attributes to provide the TAI with the certificate information for a request:

- The com.ibm.websphere.ssl.direct_connection_peer_certificates attribute contains a X509Certificate[] object of the certificate for a direct peer.
- The com.ibm.websphere.ssl.direct_connection_cipher_suite attribute contains a string object of a direct cipher suite.
- The com.ibm.websphere.webcontainer.is_direct_connection attribute contains a boolean object that indicates whether the connection was made through a web server, or was made directly to WebSphere Application Server.

See the topic *Web container request attributes* for more information about these attributes.

Trust association settings

Use this page to enable trust association, which integrates application server security and third-party security servers. More specifically, a reverse proxy server can act as a front-end authentication server while the product applies its own authorization policy onto the resulting credentials passed by the proxy server.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under Authentication, expand Web security and click **Trust association**.

When security is enabled and any of these properties change, go to the Global security panel and click **Apply** to validate the changes.

Enable trust association:

Specifies whether trust association is enabled.

Data type:	Boolean
Default:	Disable
Range:	Enable or Disable

Trust association interceptor collection

Use this page to specify trust information for reverse security proxy servers.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under Authentication, expand Web and SIP security and click **Trust association**.
3. Under Additional Properties, click **Interceptors**.

When security is enabled and any of these properties are changed, go to the Global security panel and click **Apply** to validate the changes.

Interceptor class name:

Specifies the trust association interceptor class name.

Data type
String

Trust association interceptor settings

Use this page to specify trust information for reverse security proxy servers.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under Authentication, click **Web and SIP security**.
3. Click **Trust association**.
4. Under Additional Properties, click **Interceptors > New**.

Interceptor class name:

Specifies the trust association interceptor class name.

Data type
String

Single sign-on for authentication

With single sign-on (SSO) support, web users can authenticate once when accessing both WebSphere Application Server resources, such as HTML, JavaServer Pages (JSP) files, servlets, enterprise beans, and Lotus Domino resources, such as documents in a Domino database, or accessing resources in multiple WebSphere Application Server domains.

There are various ways to accomplish SSO, with the most common in WebSphere using LTPA cookies. LTPA cookies do not require any particular client and allow SSO across different cells provide the registry and LTPA keys are the same.

There are other flavors of SSO, including Simple and Protected GSS-API Negotiation (SPNEGO), which is a way to use the token from a Kerberos login (typically Windows) to authenticate to WebSphere Application Server. This prevents the user from having to type in their userid and passwords again.

Note: In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. This function was deprecated In WebSphere Application Server 7.0. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

TAIs are also a form of single sign-on when used in combination with a Proxy server that does the front-end authentication. The TAI allows the credentials to flow to WebSphere from the Proxy server and to be used to login without the need to re-authenticate the user.

Single sign-on for authentication using LTPA cookies

With single sign-on (SSO) support, web users can authenticate once when accessing both WebSphere Application Server resources, such as HTML, JavaServer Pages (JSP) files, servlets, enterprise beans, and Lotus Domino resources, such as documents in a Domino database, or accessing resources in multiple WebSphere Application Server domains.

Application servers distributed in multiple nodes and cells can securely communicate using the Lightweight Third Party Authentication (LTPA) protocol. LTPA is intended for distributed, multiple application server and machine environments. LTPA can support security in a distributed environment through cryptography. This support permits LTPA to encrypt, digitally sign, and securely transmit authentication-related data, and later decrypt and verify the signature.

LTPA also provides the SSO feature wherein a user is required to authenticate only once in a domain name system (DNS) domain and can access resources in other WebSphere Application Server cells without getting prompted. Web users can authenticate once to a WebSphere Application Server or to a Domino server. This authentication is accomplished by configuring WebSphere Application Servers and the Domino servers to share authentication information.

Without logging in again, web users can access other WebSphere Application Servers or Domino servers in the same DNS domain that are enabled for SSO. You can enable SSO among WebSphere Application Servers by configuring SSO for WebSphere Application Server. To enable SSO between WebSphere Application Servers and Domino servers, you must configure SSO for both WebSphere Application Server and for Domino.

Prerequisites and conditions

To take advantage of support for SSO between WebSphere Application Servers or between WebSphere Application Server and a Domino server, applications must meet the following prerequisites and conditions:

- Verify that all servers are configured as part of the same DNS domain. The realm names on each system in the DNS domain are case sensitive and must match identically. For example, if the DNS domain is specified as `mycompany.com`, then SSO is effective with any Domino server or WebSphere Application Server on a host that is part of the `mycompany.com` domain, for example, `a.mycompany.com` and `b.mycompany.com`.
- Verify that all servers share the same registry.

  This registry can be either a supported Lightweight Directory Access Protocol (LDAP) directory server or, if SSO is configured between two WebSphere Application Servers, a stand-alone custom registry.Domino servers do not support stand-alone custom registries, but you can use a Domino-supported registry as a stand-alone custom registry within WebSphere Application Server.

  You can use a Domino directory that is configured for LDAP access or other LDAP directories for the registry. The LDAP directory product must have WebSphere Application Server support. Supported products include both Domino and LDAP servers, such as IBM Tivoli Directory Server. Regardless of the choice to use an LDAP or a stand-alone custom registry, the SSO configuration is the same. The difference is in the configuration of the registry.
- Define all users in a single LDAP directory. Using multiple Domino directory assistance documents to access multiple directories also is not supported.
- Enable HTTP cookies in browsers because the authentication information that is generated by the server is transported to the browser in a cookie. The cookie is used to propagate the authentication information for the user to other servers, exempting the user from entering the authentication information for every request to a different server.
- For a Domino server:
  - Domino Release 6.5.4 for iSeries and other platforms are supported.
  - A Lotus Notes® client Release 5.0.5 or later is required for configuring the Domino server for SSO.
  - You can share authentication information across multiple Domino domains.
- For WebSphere Application Server:
  - WebSphere Application Server Version 3.5 or later for all platforms are supported.

- You can use any HTTP web server that is supported by WebSphere Application Server.
- You can share authentication information across multiple product administrative domains.
- Basic authentication (user ID and password) using the basic and form-login mechanisms is supported.

  Note: Form-login mechanisms for web applications require that SSO is enabled.
- By default, WebSphere Application Server does a case-sensitive comparison for authorization. This comparison implies that a user who is authenticated by Domino matches the entry exactly (including the base distinguished name) in the WebSphere Application Server authorization table. If case sensitivity is not considered for the authorization, enable the **Ignore Case** property in the LDAP user registry settings.

Using a WebSphere Application Server API to achieve downstream web single sign-on with an LtpaToken2 cookie

You can programmatically perform downstream Single Sign On (SSO) web propagation of a Lightweight Third Party Authentication (LTPA) cookie without the need for an application to store and send user credentials.

WebSphere Application Server provides API support to propagate an LtpaToken2 cookie to downstream web single sign-on applications.

Note:

Web applications running in mid-tier WebSphere servers might need to propagate LtpaToken2 cookies on downstream web invocations. In this release of WebSphere Application Server, a new Application Programming Interface (API) is provided for application developers to programmatically perform downstream SSO without the need for an application to store and send user credentials.

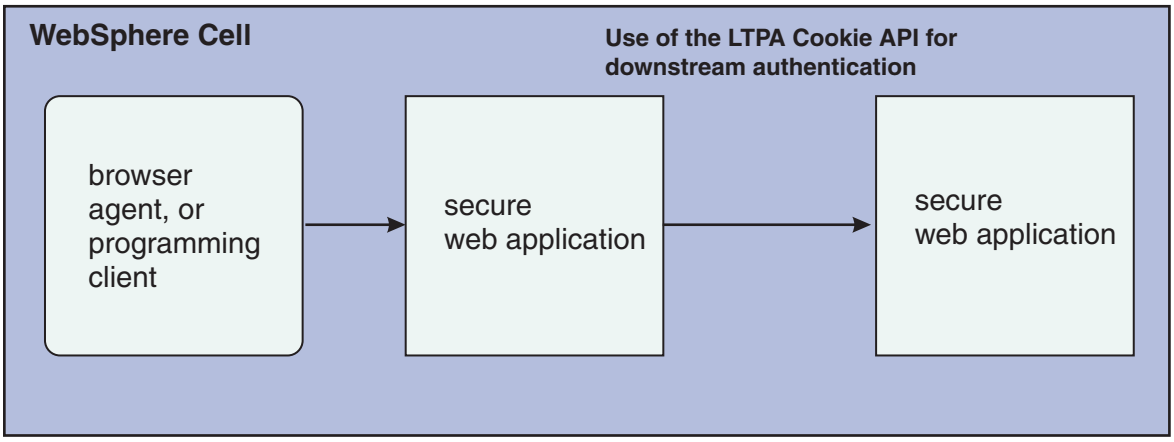


Figure 15. Use of the LTPA Cookie API for downstream authentication

This function is a public API in package com.ibm.websphere.security.WSSecurityHelper, and is defined as follows:

```
/**
   * Extracts an LTPA sso token from the subject of current
   * thread and builds a ltpa cookie out of it for use on
   * downstream web invocations.
   * When the returned value is not null use Cookie methods
   * getName() and getValue() to set the Cookie header
   * on an http request with header value of
   * Cookie.getName()=Cookie.getValue()
```

```
   *
   * @return an object of type javax.servlet.http.Cookie.
   *
   */
```

The following is an example of how you can use the new WSSecurityHelper API:

```
import javax.servlet.http.Cookie;
import com.ibm.websphere.security.WSSecurityHelper;

Cookie ltpaCookie = WSSecurityHelper.getLTPACookieFromSSOToken()
```

Subsequently, the LTPA cookie can be set on an HTTP request header. In this case, the value of the cookie header is the string:

```
ltpaCookie.getName()=ltpaCookie.getValue()
```

For example, if you use org.apache.commons.httpclient.HttpMethod to build your HTTP request, the LTPA cookie can be set as follows:

```
HttpMethod method = .;  // new your HttpMethod based on the
                        // target URL for the web application
if (ltpaCookie != null)
     method.setRequestHeader("Cookie", ltpaCookie.getName()+"="+ltpaCookie.getValue());
```

Note: You should only send LTPA cookies over SSL connections.

Note: You must check whether the LTPA cookie that is returned from calling WSSecurityHelper.getLTPACookieFromSSOToken() in the example above is not `null` before you issue any `getter` methods. Also, to successfully retrieve a LTPA cookie object, and to ensure an SSO token on the thread of execution, make sure that the user has established a successful authentication with the mid-tier server.

Note: WebSphere Application Server does not ship supporting jars for HTTP programming, such as the Apache httpclient. You must provide your own supporting functions for HTTP programming.

Enterprise Identity Mapping

Enterprise Identity Mapping (EIM) for iSeries is the OS/400 implementation of an IBM infrastructure that allows administrators and application developers to solve the problem of managing multiple user registries across their enterprise.

Most network enterprises face the problem of multiple user registries, which require each person or entity within the enterprise to have a user identity in each registry. The need for multiple user registries quickly grows into a large administrative problem that affects users, administrators, and application developers. EIM enables inexpensive solutions for easier management of multiple user registries and user identities in your enterprise.

EIM is a mechanism for mapping and associating a person or entity to the appropriate user identities in various registries throughout the enterprise. EIM provides application programming interfaces (API) for creating and managing these identity mapping relationships and provides APIs that applications use to query this information. In addition, OS/400 uses EIM and Kerberos capabilities to provide a single sign-on environment.

iSeries Navigator, the iSeries graphical user interface, provides wizards to configure and manage EIM. In addition, administrators can manage EIM relationships for user profiles through iSeries Navigator.

For more information on Enterprise Identity Mapping, see Enterprise Identity Mapping concepts.

Global single sign-on principal mapping for authentication

You can use the Java Authorization Contract for Containers (JACC) provider for Tivoli Access Manager to manage authentication to enterprise information systems (EIS) such as databases, transaction processing systems, and message queue systems that are located within the WebSphere Application Server security domain. Such authentication is achieved using the global single sign-on (GSO) principal mapper Java Authentication and Authorization Service (JAAS) login module for Java Platform, Enterprise Edition (Java EE) Connector Architecture resources.

With GSO principal mapping, a special-purpose JAAS login module inserts a credential into the subject header. This credential is used by the resource adapter to authenticate to the EIS. The JAAS login module used is configured on a per-connection factory basis. The default principal mapping module retrieves the user name and password information from XML configuration files. The JACC provider for Tivoli Access Manager bypasses the credential that is stored in the Extensible Markup Language (XML) configuration files and uses the Tivoli Access Manager global sign-on (GSO) database instead to provide the authentication information for the EIS security domain.

WebSphere Application Server provides a default principal mapping module that associates user credential information with EIS resources. The default mapping module is defined in the WebSphere Application Server administrative console on the Application login panel. To access the panel, click **Security > Global security**. Under Java Authentication and Authorization Service, click **Application logins**. The mapping module name is DefaultPrincipalMapping.

The EIS security domain user ID and password are defined under each connection factory by an authDataAlias attribute. The authDataAlias attribute does not contain the user name and password; this attribute contains an alias that refers to a user name and password pair that is defined elsewhere.

The Tivoli Access Manager principal mapping module uses the authDataAlias attribute to determine the GSO resource name and the user name that is required to perform the lookup on the Tivoli Access Manager GSO database. The Tivoli Access Manager Policy Server retrieves the GSO data from the user registry.

Tivoli Access Manager stores authentication information on the Tivoli Access Manager GSO database against a resource and user name pair.

GSO principal mapping architecture

Implementing single sign-on to minimize web user authentications

With single sign-on (SSO) support, web users can authenticate once when accessing web resources across multiple WebSphere Application Servers. Form login mechanisms for web applications require that SSO is enabled. Use this topic to configure single sign-on for the first time.

Before you begin

SSO is supported only when Lightweight Third Party Authentication (LTPA) is the authentication mechanism.

When SSO is enabled, a cookie is created containing the LTPA token and inserted into the HTTP response. When the user accesses other web resources in any other WebSphere Application Server process in the same domain name service (DNS) domain, the cookie is sent in the request. The LTPA token is then extracted from the cookie and validated. If the request is between different cells of WebSphere Application Servers, you must share the LTPA keys and the user registry between the cells for SSO to work. The realm names on each system in the SSO domain are case sensitive and must match identically.

The realm name is the same as the host name.

For the Lightweight Directory Access Protocol (LDAP) the realm name is the host:port realm name of the LDAP server. The LTPA authentication mechanism requires that you enable SSO if any of the web applications have form login as the authentication method.

Because single sign-on is a subset of LTPA, it is recommended that you read “Lightweight Third Party Authentication” on page 1413 for more information.

When you enable security attribute propagation, the following cookie is always added to the response:

LtpaToken2
LtpaToken2 contains stronger encryption and enables you to add multiple attributes to the token. This token contains the authentication identity and additional information such as the attributes that are used for contacting the original login server and the unique cache key for looking up the Subject when considering more than just the identity in determining uniqueness.

Note: The following cookie is optionally added to the response when the **Interoperability mode** flag is enabled:

LtpaToken
LtpaToken is used for inter-operating with previous releases of WebSphere Application Server. This token contains the authentication identity attribute only.

Note: LtpaToken is generated for releases prior to WebSphere Application Server Version 5.1.1. LtpaToken2 is generated for WebSphere Application Server Version 5.1.1 and beyond.

Table 87. LTPA token types. This table describes the LTPA token types.

Token type	Purpose	How to specify
LtpaToken2 only	This is the default token type. It uses the AES-CBC-PKCS5 padding encryption strength (128-bit key size). This token is stronger than the older LtpaToken used prior to WebSphere Application Server Version 6.02. This is the recommended option when interoperability with older releases is not necessary.	Disable the **Interoperability mode** option in the SSO configuration panel within the administrative console. To access this panel, complete the following steps: 1. Click **Security > Global security**. 2. Under Web security, click **Single sign-on (SSO)**.
LtpaToken and LtpaToken2	Use to interoperate with releases prior to WebSphere Application Server Version 5.1.1. The older LtpaToken cookie is present along with the new LtpaToken2 cookie. Provided the LTPA keys are correctly shared, you should be able to interoperate with any version of WebSphere using this option.	Enable the **Interoperability mode** option in the SSO configuration panel within the administrative console. To access this panel, complete the following steps: 1. Click **Security > Global security**. 2. Under Web security, click **Single sign-on (SSO)**.

About this task

The following steps are required to configure SSO for the first time.

Procedure

1. Open the administrative console.

   Type `http://server_name:port_number/ibm/console` to access the administrative console in a web browser.

   Port 9060 is the default port number for accessing the administrative console. During installation, however, you might have specified a different port number. Use the appropriate port number.
2. Click **Security > Global security**.
3. Under Web security, click **Single sign-on (SSO)**.

4. Click the **Enabled** option if SSO is disabled. After you click the **Enabled** option, make sure that you complete the remaining steps to enable security.
5. Click **Requires SSL** if all of the requests are expected to use HTTPS.
6. Enter the fully qualified domain names in the **Domain name** field where SSO is effective. If you specify domain names, they must be fully qualified. If the domain name is not fully qualified, WebSphere Application Server does not set a domain name value for the LtpaToken cookie and SSO is valid only for the server that created the cookie.

   When you specify multiple domains, you can use the following delimiters: a semicolon (**;**), a space ( ), a comma (**,**), or a pipe (**|**). WebSphere Application Server searches the specified domains in order from left to right. Each domain is compared with the host name of the HTTP request until the first match is located. For example, if you specify `ibm.com®; austin.ibm.com` and a match is found in the ibm.com domain first, WebSphere Application Server does not continue to search for a match in the austin.ibm.com domain. However, if a match is not found in either the ibm.com or austin.ibm.com domains, then WebSphere Application Server does not set a domain for the LtpaToken cookie.

Table 88. Values to configure the Domain name field.

This table describes the values to configure the Domain name field.

Domain name value type	Example	Purpose
Blank		The domain is not set. This causes the browser to set the domain to the request host name. The sign-on is valid on that single host only.
Single domain name	austin.ibm.com	If the request is to a host within the configured domain, the sign-on is valid for all hosts within that domain. Otherwise, it is valid on the request host name only.
UseDomainFromURL	UseDomainFromURL	If the request is to a host within the configured domain, the sign-on is valid for all hosts within that domain. Otherwise, it is valid on the request host name only.
Multiple domain names	austin.ibm.com;raleigh.ibm.com	The sign-on is valid for all hosts within the domain of the request host name.
Multiple domain names and UseDomainFromURL	• austin.ibm.com;raleigh.ibm.com; UseDomainFromURL	The sign-on is valid for all hosts within the domain of the request host name.

   If you specify the UseDomainFromURL, WebSphere Application Server sets the SSO domain name value to the domain of the host that makes the request. For example, if an HTTP request comes from server1.raleigh.ibm.com, WebSphere Application Server sets the SSO domain name value to `raleigh.ibm.com` .

   Tip: The value, `UseDomainFromURL`, is case insensitive. You can type `usedomainfromurl` to use this value.

   For more information, see “Single sign-on settings” on page 1501.
7. Optional: Enable the **Interoperability mode** option if you want to support SSO connections in WebSphere Application Server version 5.1.1 or later to interoperate with previous versions of the application server.

   This option sets the old-style LtpaToken token into the response so it can be sent to other servers that work only with this token type. Otherwise, only the LtpaToken2 token is added to the response.

   If performance is a consideration, and you are only connecting to Version 6.1 or later servers that and are not running products that depend on the LtpaToken, do not enable Interoperability mode. When Interoperability mode is not enabled, an LtpaToken is not returned in a response.
8. Optional: Enable the **Web inbound security attribute propagation** option if you want information added during the login at a specific front-end server to propagate to other front-end servers. The SSO token does not contain any sensitive attributes, but does understand where the original login server exists in cases where it needs to contact that server to retrieve serialized information. For more information, see “Security attribute propagation” on page 1544.

Important: If the following statements are true, it is recommended that you disable the **Web inbound security attribute propagation** option for performance reasons:

- You do not have any specific information added to the Subject during a login that cannot be obtained at a different front-end server.
- You did not add custom attributes to the PropagationToken token using WSSecurityHelper application programming interfaces (APIs).

If you find that you are missing custom information in the Subject, re-enable the **Web inbound security attribute propagation** option to see if the information is propagated successfully to other front-end application servers.

The following two custom properties might help to improve performance when security attribute propagation is enabled:

- **com.ibm.CSI.propagateFirstCallerOnly**

  The default value of this property is `true`. When this custom property is set to `true` the first caller in the propagation token that stays on the thread is logged when security attribute propagation is enabled. When this property is set to `false`, all of the caller switches are logged, which can affect performance.
- **com.ibm.CSI.disablePropagationCallerList**

  When this custom property is set to `true` the ability to add a caller or host list in the propagation token is completely disabled. This function is beneficial when the caller or host list in the propagation token is not needed in the environment.

9. Click **OK**.

What to do next

For the changes to take effect, save, stop, and restart all the product servers.

Single sign-on for HTTP requests using SPNEGO web authentication

You can securely negotiate and authenticate HTTP requests for secured resources in WebSphere Application Server by using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) as the web authentication service for WebSphere Application Server.

Note: In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. This function was deprecated in WebSphere Application Server Version 7.0. SPNEGO web authentication has taken its place to provide the following enhancements:

- You can configure and enable SPNEGO web authentication and filters on WebSphere Application Server by using the administrative console.
- Dynamic reload of SPNEGO is provided without the need to stop and restart WebSphere Application Server.
- Fallback to an application login method is provided if the SPNEGO web authentication fails.
- SPNEGO can be customized at the WebSphere security domain level. Read about “Multiple security domains” on page 1222 for more information.

You can enable either SPNEGO TAI or SPNEGO Web Authentication but not both.

The following sections describe SPNEGO web authentication in more detail:

- “What is SPNEGO?” on page 1445
- “The benefits of SPNEGO web authentication” on page 1446
- “SPNEGO web authentication in a single Kerberos realm” on page 1446
- “SPNEGO web authentication in a trusted Kerberos realm” on page 1447

- “Support information for SPNEGO web authentication with a Java client using the HTTP protocol” on page 1448
- “Support information for SPNEGO web authentication with a browser client” on page 1448
- “Setting up SPNEGO as the web authentication mechanism for WebSphere Application Server” on page 1448

What is SPNEGO?

SPNEGO is a standard specification defined in The Simple and Protected GSS-API Negotiation Mechanism (IETF RFC 2478).

When WebSphere Application Server global and application security are enabled, and SPNEGO web authentication is enabled, SPNEGO is initialized when processing a first inbound HTTP request. The web authenticator component then interacts with SPNEGO, which is defined and enabled in the security configuration repository. When the filter criteria is met, SPNEGO is responsible for authenticating access to the secured resource that is identified in the HTTP request.

In addition to WebSphere Application Server security runtime services, some external components are required to enable the operation of SPNEGO. These external components include:
- A client application, for example, Microsoft .NET, or web service and J2EE client that supports the SPNEGO web authentication mechanism, as defined in IETF RFC 2478. Microsoft Internet Explorer Version 5.5 or later and Mozilla Firefox Version 1.0 are browser examples. Any browser must be configured to use the SPNEGO web authentication mechanism. For more information on performing this configuration, see Configuring the client browser to use SPNEGO.

The authentication of HTTP requests is triggered by the requestor (the client-side), which generates a SPNEGO token. WebSphere Application Server receives this token. Specifically, the SPNEGO web authentication decodes and retrieves the requester's identity from the SPNEGO token. The identity is used to establish a secure context between the requester and the application server.

SPNEGO web authentication is a server-side solution in WebSphere Application Server. Client-side applications are responsible for generating the SPNEGO token for use by SPNEGO web authentication. The requester's identity in the WebSphere Application Server security registry must be identical to the identity that the SPNEGO web authentication retrieves. An identical match does occur when Microsoft Windows Active Directory server is the Lightweight Directory Access Protocol (LDAP) server that is used in WebSphere Application Server. A custom login module is available as a plug-in to support custom mapping of the identity from the Active Directory to the WebSphere Application Server security registry.

Read about Mapping of a client Kerberos principal name to the WebSphere user registry ID for more information about using this custom login module.

WebSphere Application Server validates the identity against its security registry. If the validation is successful, the client Kerberos ticket and GSS delegation credential are retrieved and placed in the client subject, which then produces a Lightweight Third Party Authentication (LTPA) security token. It then places and returns a cookie to the requester in the HTTP response. Subsequent HTTP requests from this same requester to access additional secured resources in WebSphere Application Server use the LTPA security token previously created to avoid repeated login challenges.

The web administrator has access to the following SPNEGO security components and associated configuration data, as shown in the following figure:

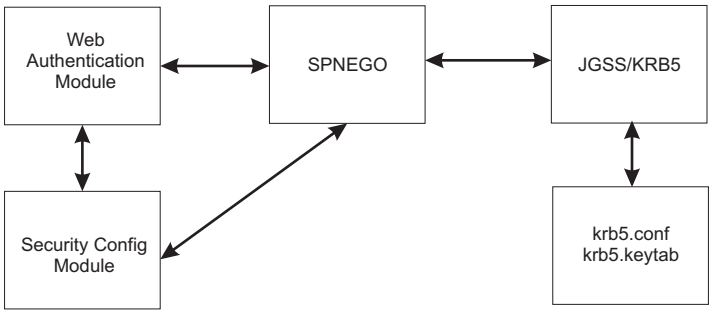


Figure 16. SPNEGO web authentication and security configuration elements

The benefits of SPNEGO web authentication

The benefits of having WebSphere Application Server use SPNEGO as the web authentication service for WebSphere Application Server include the following:

- The cost of administering a large number of ids and passwords is reduced.
- A secure and mutually authenticated transmission of security credentials from the web browser or Microsoft .NET clients is established.
- Interoperability with web services and Microsoft .NET, or web service applications that use SPNEGO authentication at the transport level is achieved.
- With Kerberos authentication support, SPNEGO web authentication can provide an end-to-end SPNEGO to Kerberos solution and preserve the Kerberos credential from the client.

SPNEGO web authentication in a single Kerberos realm

SPNEGO web authentication is supported in a single Kerberos realm. The challenge-response handshake process is shown in the following figure:

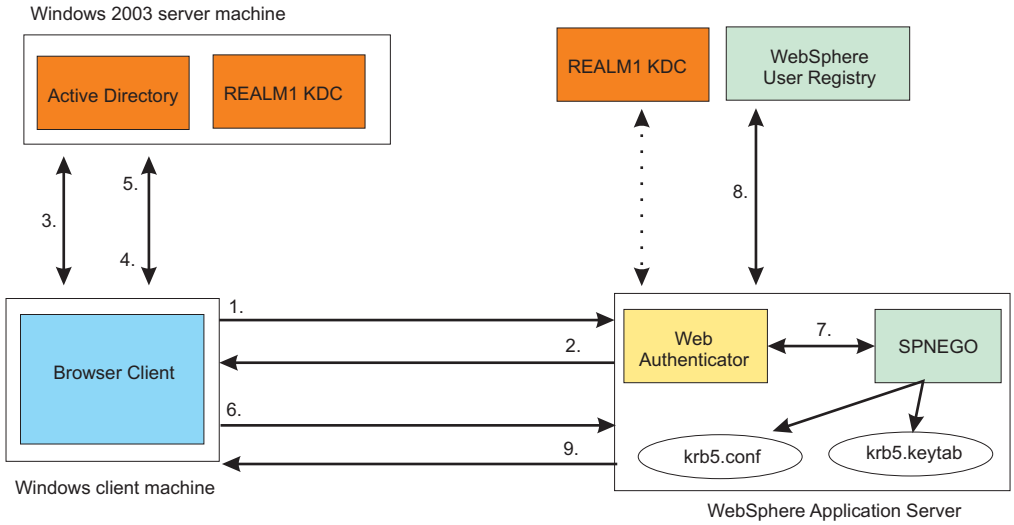


Figure 17. SPNEGO web authentication in a single Kerberos realm

In the figure above, the following events occur:

1. The client sends an HTTP/Post/Get/Web-Service request to WebSphere Application Server.
2. WebSphere Application Server returns `HTTP 401` Authenticate/Negotiate.
3. The client obtains a Ticket Granting Ticket (TGT).

4. The client requests a Service Ticket (TGS_REQ).
5. The client obtains a Service Ticket (TGS_REP).
6. The client sends HTTP/Post/Get/Web-Service and an authorization SPNEGO token to WebSphere Application Server.
7. WebSphere Application Server validates the SPNEGO token. If the validation is successful, it retrieves the user ID and the GSS delegation credential from the SPNEGO token. Create a KRBAuthnToken with a client Kerberos credential.
8. WebSphere Application Server validates the user ID with the WebSphere user registry and creates an LTPA token.
9. WebSphere Application Server returns `HTTP 200`, content and the LTPA token to the client.

Note: Other clients (for example, web services, .NET and J2EE) that support SPNEGO do not have to follow the challenge-response handshake process as shown above. Those clients can obtain a ticket-granting ticket (TGT) and a Kerberos service ticket for the target server, create a SPNEGO token, insert it in the HTTP header, and then follow the normal process for creating an HTTP request.

SPNEGO web authentication in a trusted Kerberos realm

SPNEGO web authentication is also supported in a trusted Kerberos realm. The challenge-response handshake process is shown in the following figure:

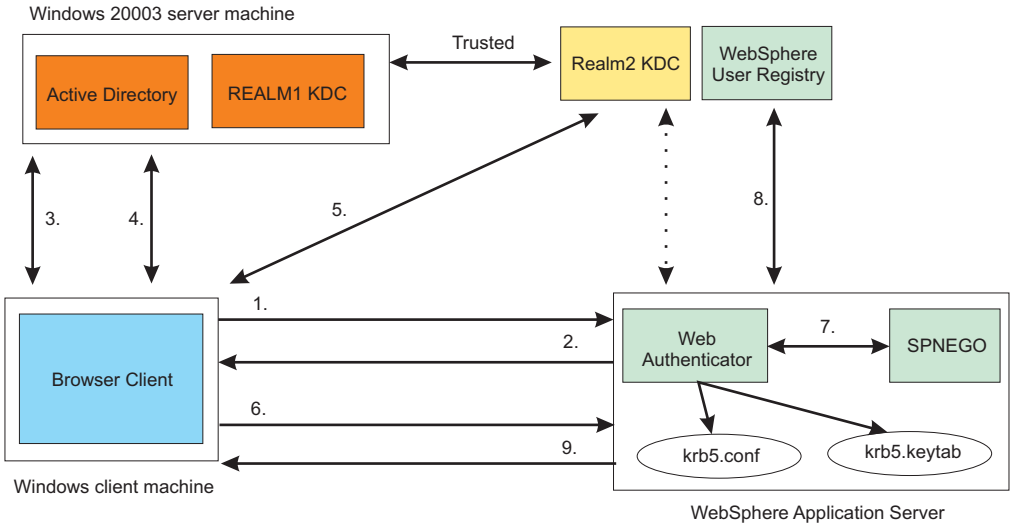


Figure 18. SPNEGO web authentication in a trusted Kerberos realm

In the figure above, the following events occur:

1. The client sends an HTTP/Post/Get/Web-Service request to WebSphere Application Server.
2. WebSphere Application Server returns `HTTP 401` Authenticate/Negotiate
3. The client obtains a Ticket Granting Ticket (TGT).
4. The client requests a cross realm ticket (TGS_REQ) for `REALM2` from the `REALM1` KDC.
5. The client uses the cross-realm ticket from step 4 to request a Service Ticket from the `REALM2` KDC.
6. The client sends HTTP/Post/Get/Web-Service and an authorization SPNEGO token to WebSphere Application Server.

7. WebSphere Application Server validates the SPNEGO token. If the validation is successful, it retrieves the user ID and the GSS delegation credential from the SPNEGO token. Create a KRBAuthnToken with a client Kerberos credential.
8. WebSphere Application Server validates the user ID with the WebSphere user registry and creates an LTPA token.
9. WebSphere Application Server returns `HTTP 200`, content and the LTPA token to the client.

In the trusted Kerberos realms environment, be aware of the following:

- The Kerberos trusted realm setup must be done on each of the Kerberos KDCs. See your Kerberos Administrator and User's guide for more information about how to set up Kerberos trusted realms.
- The Kerberos client principal name from the SPNEGO token might not exist in the WebSphere user registry; the Kerberos principal mapping to the WebSphere user registry might require it.

  Read about Mapping of a client Kerberos principal name to the WebSphere user registry ID for more information.

Support information for SPNEGO web authentication with a Java client using the HTTP protocol

The following scenarios are supported:

- Domain trust within the same forest
- External domain trust directly between domains within different forests.
- Kerberos realm trust

The following scenarios are not supported:

- Cross-forest trust
- Forest external trust

Support information for SPNEGO web authentication with a browser client

The following scenarios are supported:

- Cross-forest trusts
- Domain trust within the same forest
- Kerberos realm trust

The following scenarios are not supported:

- Forest external trusts
- Domain external trusts

Setting up SPNEGO as the web authentication mechanism for WebSphere Application Server

Before you set up SPNEGO web authentication in the administrative console or by using **wsadmin** commands, you must perform the steps as listed in “Creating a single sign-on for HTTP requests using SPNEGO Web authentication” to set up SPNEGO web authentication for WebSphere Application Server.

Note: SPNEGO web authentication on the server side must be done by the system administrator. The Kerberos keytab file must be protected.

Creating a single sign-on for HTTP requests using SPNEGO Web authentication

Creating single sign-ons for HTTP requests using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) web authentication for WebSphere Application Server requires the performance of

several distinct, yet related functions that when completed, allow HTTP users to log in and authenticate to the Microsoft domain controller only once at their desktop and to receive automatic authentication from the WebSphere Application Server.

Before you begin

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. This function was deprecated in WebSphere Application Server Version 7.0. SPNEGO web authentication has taken its place to provide the following enhancements:

- You can configure and enable SPNEGO web authentication and filters on the WebSphere Application Server server side by using the administrative console.
- Dynamic reload of SPNEGO is provided without the need to stop and restart the WebSphere Application Server server.
- Fallback to an application login method is provided if the SPNEGO web authentication fails.

You can enable either SPNEGO TAI or SPNEGO Web Authentication but not both.

Read about “Single sign-on for HTTP requests using SPNEGO web authentication” on page 1444 for a better understanding of what SPNEGO Web Authentication is and how it is supported in this version of WebSphere Application Server.

Before starting this task, complete the following checklist:

- The domain member has users who can log on to the domain. Specifically, you need to have a functioning Microsoft Windows active directory domain that includes:
  - Domain controller
  - Client workstation
  - Users who can login to the client workstation
- A server platform with WebSphere Application Server running and application security enabled.
- Users on the active directory must be able to access WebSphere Application Server protected resources using a native WebSphere Application Server authentication mechanism.
- The domain controller and the host of WebSphere Application Server should have the same local time.
- Ensure the clock on clients, Microsoft Active Directory and WebSphere Application Server are synchronized to within five minutes.
- Be aware that client browsers must be SPNEGO enabled, which you perform on the client application machine (with details explained in procedure 4, "Configure the client application on the client application machine").

About this task

The objective of this machine arrangement is to permit users to successfully access WebSphere Application Server resources without having to authenticate again and thus achieve Microsoft Windows desktop single sign-on capability.

Configuring the members of this environment to establish Microsoft Windows single sign-on involves specific activities that are performed on three distinct machines:

- A Microsoft Windows server running the Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC).
- A Microsoft Windows domain member (client application), such as a browser or Microsoft .NET client.
- A server platform with WebSphere Application Server running.

Continue with the following steps to create a single sign-on for HTTP requests using SPNEGO Web authentication:

Procedure

1. Create a Kerberos service principal (SPN) and keytab file on your Microsoft domain controller machine You must configure your domain controller machine to create single sign-ons for HTTP requests using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) web authentication for WebSphere® Application Server. Configure the Microsoft Windows Server running the Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC).

   Read the Configuring your domain controller machine to create single sign-ons for HTTP requests using SPNEGO article for more information.
2. Create a Kerberos configuration file The IBM implementation of the Java Generic Security Service (JGSS) and KRB5 require a Kerberos configuration file (**krb5.conf** or **krb5.ini**) on each node or Java virtual machine (JVM). In this release of WebSphere Application Server, this configuration file should be placed in the `config/cells/<cell_name>` directory so that all application servers can access this file. If you do not have a Kerberos configuration file, use a **wsadmin** command to create one.

   Read the Creating a Kerberos configuration article for more information.
3. Configure and enable SPNEGO web authentication using the administrative console on your WebSphere Application Server machine You can enable and configure the Simple and Protected GSS-API Negotiation (SPNEGO) as the web authenticator for the application server by using the administrative console on the WebSphere Application Server machine.

   Read the Enabling and configuring SPNEGO web authentication using the administrative console article for more information.
4. Configure the client application on the client application machine Client-side applications are responsible for generating the SPNEGO token. You begin this configuration process by configuring your web browser to use SPNEGO authentication.

   Read the Configuring the client browser to use SPNEGO article for more information.
5. Create SPNEGO tokens for J2EE, .NET, Java, web service clients for HTTP requests (optional) You can create a Simple and Protected GSS-API Negotiation (SPNEGO) token for your applications and insert this token into the HTTP headers to authenticate to the WebSphere Application Server.

   Read the Creating SPNEGO tokens for J2EE, .NET, Java, web service clients for HTTP requests article for more information.

Creating a single sign-on for HTTP requests using the SPNEGO TAI (deprecated)

Creating single sign-ons for HTTP requests using the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) for WebSphere Application Server requires the performance of several distinct, yet related functions that when completed, allow HTTP users to log in and authenticate only once at their desktop and receive automatic authentication from the WebSphere Application Server.

Before you begin

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

Before starting this task, complete the following checklist:

- The domain member has users who can log on to the domain. Specifically, you need to have a functioning Microsoft Windows active directory domain that includes:

  - Domain controller
  - Client workstation
  - Users who can login to the client workstation
- A server platform with WebSphere Application Server running and application security enabled.
- Users on the active directory must be able to access WebSphere Application Server protected resources using a native WebSphere Application Server authentication mechanism.
- The domain controller and the host of WebSphere Application Server should have the same local time.
- Ensure the clock on clients, Microsoft Active Directory and WebSphere Application Server are synchronized to within five minutes.
- Be aware that client browsers have to be SPNEGO enabled, which you perform on the client application machine (with details explained in step 2 of this task).

About this task

The objective of this machine arrangement is to permit users to successfully access WebSphere Application Server resources without having to reauthenticate and thus achieve Microsoft Windows desktop single sign-on capability.

Configuring the members of this environment to establish Microsoft Windows single sign-on involves specific activities that are performed on three distinct machines:

- Microsoft Windows Server running the Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC)
- A Microsoft Windows domain member (client application), such as a browser or Microsoft .NET client.
- A server platform with WebSphere Application Server running.

Perform the following steps on the indicated machines to create single sign-on for HTTP requests using SPNEGO

Procedure

1. **Domain Controller Machine** - Configure the Microsoft Windows Server running the Active Directory Domain Controller and associated Kerberos Key Distribution Center (KDC) This configuration activity has the following steps:
   - Create a user account for the WebSphere Application Server in a Microsoft Active Directory. This account will be eventually mapped to the Kerberos service principal name (SPN).
   - On the Microsoft Active Directory machine where the Kerberos key distribution center (KDC) is active, map the user account to the Kerberos service principal name (SPN). This user account represents the WebSphere Application Server as being a Kerberize'd service with the KDC. Use the **setspn** command to map the Kerberos service principal name to a Microsoft user account. The topic, “Creating a Kerberos service principal and keytab file that is used by the WebSphere Application Server SPNEGO TAI (deprecated)” on page 1455 has more details about using the **setspn** command.
   - Create the Kerberos keytab file and make it available to WebSphere Application Server. Use the **ktpass** tool to create the Kerberos keytab file (`krb5.keytab`). The topic, “Creating a Kerberos service principal and keytab file that is used by the WebSphere Application Server SPNEGO TAI (deprecated)” on page 1455 has more details about using the **ktpass** command. to create the keytab file.

     Note: You make the keytab file available to WebSphere Application Server by copying the `krb5.keytab` file from the Domain Controller (LDAP machine) to the WebSphere Application Server machine. See “Using the ktab command to manage the Kerberos keytab file” on page 1459 for more details.

   Important: Your domain controller operations must lead to the following results:

- A user account is created in the Microsoft Active Directory and mapped to a Kerberos service principal name.
- A Kerberos keytab file (`krb5.keytab`) is created and made available to the WebSphere Application Server. The Kerberos keytab file contains the Kerberos service principal keys WebSphere Application Server uses to authenticate the user in the Microsoft Active Directory and the Kerberos account.

2. **Client Application Machine** - Configure the client application. Client-side applications are responsible for generating the SPNEGO token for use by the SPNEGO TAI. You begin this configuration process by configuring your web browser to use SPNEGO authentication. See “Configuring the client browser to use SPNEGO TAI (deprecated)” on page 1473 for the detailed steps required for your browser.
3. **WebSphere Application Server Machine** - Configure and enable the Application Server and the associated SPNEGO TAI by performing the following tasks:
   - Ensure that LTPA is enabled. See Configuring the Lightweight Third Party Authentication mechanism for more details.
   - Enable the SPNEGO TAI. See “Configuring WebSphere Application Server and enabling the SPNEGO TAI (deprecated)” on page 1460 for more details.
   - Create SPNEGO TAI properties using either the wsadmin command task or the administrative console.
     – For using the wsadmin command task, see
       - SpnegoTAICommands group for the AdminTask object (deprecated)
     – For using the administrative console, see “Configuring WebSphere Application Server and enabling the SPNEGO TAI (deprecated)” on page 1460 for more details.
   - Configure JVM properties and enable the SPNEGO TAI in Application Server in which it is defined. See “Configuring JVM custom properties, filtering HTTP requests, and enabling SPNEGO TAI in WebSphere Application Server (deprecated)” on page 1475 or “Enabling the SPNEGO TAI as JVM custom property using scripting (deprecated)” on page 1476 for more details.
   - Install the Kerberos keytab file (created in step 1) on the WebSphere Application Server machine. “Creating a Kerberos service principal and keytab file that is used by the WebSphere Application Server SPNEGO TAI (deprecated)” on page 1455 provides the details.
   - Create a basic Kerberos configuration file (krb5.ini or krb5.conf). See The Kerberos configuration file for details.
   - Map the client Kerberos principal name to the WebSphere user registry ID, but only if the WebSphere Application Server does not use Micorsoft Active Directory. See “Mapping Kerberos client principal name to WebSphere user registry ID for SPNEGO TAI (deprecated)” on page 1480 for more details.
4. Optional: **Using a remote HTTP server** - To use a remote server, you must complete the following steps, which assume that you have already configured the JVM properties and enabled the SPNEGO TAI in the Application Server in which it is defined (as described in the previous three steps).
   a. Complete the steps in “Creating a Kerberos service principal and keytab file that is used by the WebSphere Application Server SPNEGO TAI (deprecated)” on page 1455 for the remote proxy server.
   b. Merge the previous keytab file created in step 1 with the keytab file created in step 4a. See “Using the ktab command to manage the Kerberos keytab file” on page 1459 for more information.
   c. Create the SPN for the remote proxy server using the `addSpnegoTAIProperties wsadmin` command task. For more information, see SpnegoTAICommands group for the AdminTask object (deprecated).
   d. Restart the WebSphere Application Server.

Single sign-on for HTTP requests using SPNEGO TAI (deprecated):

WebSphere Application Server provides a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources in WebSphere Application Server.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

Read about “Creating a single sign-on for HTTP requests using SPNEGO Web authentication” on page 1448 for more information.

SPNEGO is a standard specification defined in The Simple and Protected GSS-API Negotiation Mechanism (IETF RFC 2478).

When WebSphere Application Server administrative security is enabled, the SPNEGO TAI is initialized. While processing inbound HTTP requests, the web authenticator component interacts with the SPNEGO TAI, which is defined and enabled in the security configuration repository. One interceptor is selected and is responsible for authenticating access to the secured resource that is identified in the HTTP request.

Important: The use of TAIs is an optional feature. If no TAI is selected, the authentication process continues normally.

HTTP users log in and authenticate only once at their desktop and are subsequently authenticated (internally) with WebSphere Application Server. The SPNEGO TAI is invisible to the end-user of WebSphere applications. The SPNEGO TAI is only visible to the web administrator who is responsible for ensuring a proper configuration, capacity, and maintenance of the web environment.

In addition to WebSphere Application Server security runtime services, some external components are required to completely enable operation of the SPNEGO TAI. The external components include:

- A client application, for example, a browser or Microsoft .NET client, that supports the SPNEGO authentication mechanism, as defined in IETF RFC 2478. Microsoft Internet Explorer Version 5.5 or later and Mozilla Firefox Version 1.0 are browser examples. Any browser needs to be configured to use the SPNEGO mechanism. For more information on performing this configuration, see “Configuring the client browser to use SPNEGO TAI (deprecated)” on page 1473.

The authentication of HTTP requests is triggered by the requestor (the client-side), which generates a SPNEGO token. WebSphere Application Server receives this token and validates trust between the requester and WebSphere Application Server. Specifically, the SPNEGO TAI decodes and retrieves the requester's identity from the SPNEGO token. The identity is used to establish a secure context between the requester and the application server.

Remember: The SPNEGO TAI is a server-side solution in WebSphere Application Server. Client-side applications are responsible for generating the SPNEGO token for use by the SPNEGO TAI. The requester's identity in WebSphere Application Server security registry must be identical to that identity the SPNEGO TAI retrieves. An identical match does occur when Microsoft Windows Active Directory server is the Lightweight Directory Access Protocol (LDAP) server that is used in WebSphere Application Server. A custom login module is available as a plug-in to support custom mapping of the identity from the Active Directory to the WebSphere Application Server security registry. See “Mapping Kerberos client principal name to WebSphere user registry ID for SPNEGO TAI (deprecated)” on page 1480 for details on using this custom login module.

WebSphere Application Server validates the identity against its security registry and, if the validation is successful, produces a Lightweight Third Party Authentication (LTPA) security token and places and returns a cookie to the requester in the HTTP response. Subsequent HTTP requests from this same requester to access additional secured resources in WebSphere Application Server use the LTPA security token previously created, to avoid repeated login challenges.

The challenge-response handshake process is illustrated in the following graphic:

Figure 19. HTTP request processing, WebSphere Application Server - SPNEGO TAI

The SPNEGO TAI can be enabled for all or for selected WebSphere Application Servers in a WebSphere Application Server cell configuration. Also, the behavior of each SPNEGO TAI instance is controlled by custom configuration properties that are used to identify, for example, the criteria used to filter HTTP requests, such as the host name and security realm name used to construct the Kerberos Service Principal Name (SPN). For more information regarding establishing and setting the SPNEGO TAI custom configuration properties, see the following topics:

- Setting up the Kerberos configuration properties. See The Kerberos configuration file.
- Setting or adjusting the SPNEGO TAI custom properties. See “SPNEGO TAI custom properties configuration (deprecated)” on page 1469.
- Adjusting the SPNEGO TAI filter settings. See “Configuring JVM custom properties, filtering HTTP requests, and enabling SPNEGO TAI in WebSphere Application Server (deprecated)” on page 1475
- Using the custom login module to map the identity from the Active Directory to the WebSphere Application Server registry. See Mapping user Ids from client to server for SPNEGO.
- Setting the major and additional Java virtual machine (JVM) custom properties. See “SPNEGO TAI JVM configuration custom properties (deprecated)” on page 1477

The web administrator has access to the following SPNEGO TAI security components and associated configuration data, as illustrated in the following graphic.

Figure 20. SPNEGO TAI security and configuration elements

- The web authentication module and the Lightweight Third Party Authentication (LTPA) mechanism provide the plug-in runtime framework for trust association interceptors. See Configuring the Lightweight Third Party Authentication mechanism for more detail is configuring the LTPA mechanism for use with the SPNEGO TAI.
- The Java Generic Security Service (JGSS) provider is included in the Java SDK (*app_server_root*/java/endorsed/ibmjgssprovider.jar) and used to obtain the Kerberos security context and credentials that are used for authentication. IBM JGSS 1.0 is a Java Generic Security Service Application Programming Interface (GSSAPI) framework with Kerberos V5 as the underlying default security mechanism. GSSAPI is a standardized abstract interface under which can be plugged different security mechanisms based on private-key, public-key and other security technologies. GSSAPI shields secure applications from the complexities and peculiarities of the different underlying security mechanisms. GSSAPI provides identity and message origin authentication, message integrity, and message confidentiality. For more information, see JGSS.
- The Kerberos configuration properties (krb5.conf or krb5.ini) and Kerberos encryption keys (stored in a Kerberos keytab file) are used to establish secure mutual authentication.

  The Kerberos key table manager (Ktab), which is part of JGSS, allows you to manage the principal names and service keys stored in a local Kerberos keytab file. Principal name and key pairs listed in the

Kerberos keytab file allow services running on a host to authenticate themselves to the Kerberos Key Distribution Center (KDC). Before a server can use Kerberos, a Kerberos keytab file must be initialized on the host that runs the server.

"Using the ktab command to manage the Kerberos keytab file" on page 1459 highlights the Kerberos configuration requirements for the SPNEGO TAI as well as the use of Ktab.

- The SPNEGO provider supplies the implementation of the SPNEGO authentication mechanism, located at *app_server_root*/java/ext/ibmspnego.jar.
- The custom configuration properties control the runtime behavior of the SPNEGO TAI. Configuration operations are performed with the administrative console or scripting facilities. Refer to "SPNEGO TAI custom properties configuration (deprecated)" on page 1469 for more information about these custom configuration properties.
- Java virtual machine (JVM) custom properties control diagnostic trace information for problem determination of the JGSS security provider and use of the property reload feature."SPNEGO TAI JVM configuration custom properties (deprecated)" on page 1477 describes these JVM custom properties

The benefits of having WebSphere Application Server use the SPNEGO TAI include:

- The cost of administering a large number of ids and passwords is reduced.
- A secure and mutually authenticated transmission of security credentials from the web browser or Microsoft .NET clients is established.
- Interoperability with web services and Microsoft .NET applications that use SPNEGO authentication at the transport level is achieved.

Using the SPNEGO TAI in your WebSphere Application Server environment requires planning then implementation. See "Single sign-on capability with SPNEGO TAI - checklist (deprecated)" on page 1484 in planning for SPNEGO TAI. Implementing the use of the SPNEGO TAI is divided into the following areas of responsibility:

End browser user
The end user must configure the web browser or Microsoft .NET application to issue HTTP requests that are processed by the SPNEGO TAI.

Web administrator
The web administrator is responsible for configuring the SPNEGO TAI of WebSphere Application Server to respond to HTTP requests of the client.

WebSphere Application Server administrator
The WebSphere Application Server administrator is responsible for configuring WebSphere Application Server and the SPNEGO TAI for optimum installation performance.

See "Creating a single sign-on for HTTP requests using the SPNEGO TAI (deprecated)" on page 1450 for an explanation of the tasks required to use the SPNEGO TAI and how the responsible party performs these tasks.

Creating a Kerberos service principal and keytab file that is used by the WebSphere Application Server SPNEGO TAI (deprecated):

You perform this configuration task on the Microsoft Active Directory domain controller machine. This task is a necessary part of preparing to process single sign on browser requests to WebSphere Application Server and thee SPNEGO trust association interceptor (TAI).

Before you begin

You need to have a running domain controller and at least one client machine in that domain.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

About this task

This task is performed on the active directory domain controller machine. Complete the following steps to ensure that the Microsoft Windows Server that is running the active directory domain controller is configured properly to the associated key distribution center (KDC). For information on the supported Microsoft Windows Servers, see the System Requirements for WebSphere Application Server Version 8.0 on Windows.

Procedure

1. Create a user account in the Microsoft Active Directory for the WebSphere Application Server.

   Click **Start->Programs->Administrative Tools->Active Directory Users and Computers**

   Use the name for the WebSphere Application Server. For example, if the Application Server you are running on the WebSphere Application Server machine is called `myappserver.austin.ibm.com`, create a new user in Active Directory called `myappserver`.

   Important: Do not select "User must change password at next logon."

   Important: Make sure that you do not have the computer name `myappserver` under Computers and Domain Controllers (You check for this condition as illustrated below.). If you already have a computer name `myappserver`, then you need to create a different user account name.
   - Goto **Start -> Programs -> Administrative Tools -> Active Directory Users and Computers->Computers**
   - Goto **Start -> Programs -> Administrative Tools -> Active Directory Users and Computers->Domain Controllers**

2. Use the **setspn** command to map the Kerberos service principal name, `HTTP/<host name>`, to a Microsoft user account. An example of **setspn** usage is as follows:

   ```
   C:\Program Files\Support Tools>
   setspn -A HTTP/myappserver.austin.ibm.com myappserver
   ```

   Note: There may already be some SPNs related to the Microsoft Windows hosts that have been added to the domain. You can display those that exist by using the `setspn -L` command, but you still have to add an HTTP SPN for WebSphere Application Server. For example, `setspn -L myappserver` would list the SPNs.

   Important: Make sure that you do not have the same SPNs mapping to more than one Microsoft user account. If you map the same SPN to more than one user account, the web browser client can send a NTLM instead of SPNEGO token to WebSphere Application Server.

   More information about the **setspn** command can be found here, Windows 2003 Technical Reference (setspn command)

3. Create the Kerberos keytab file and make it available to WebSphere Application Server. Use the ktpass command to create the Kerberos keytab file (krb5.keytab).

   Use the **ktpass** tool from the Windows Server toolkit to create the Kerberos keytab file for the service principal name (SPN). Use the latest version of the **ktpass** tool that matches the Windows server level that you are using. For example, use the Windows 2003 version of the tool for a Windows 2003 server.

   To determine the appropriate parameter values for the **ktpass** tool, run the `ktpass -?` command from the command line. This command lists whether the **ktpass** tool, which corresponds to the particular

operating system, uses the `-crypto RC4-HMAC` or `-crypto RC4-HMAC-NT` parameter value. To avoid warning messages from the toolkit, you must specify the `-ptype KRB5_NT_PRINCIPAL` parameter value.

The Windows 2003 server version of the **ktpass** tool supports the encryption type, RC4-HMAC, and Single data encryption standard (DES). For more information about the **ktpass** tool, see Windows 2003 Technical Reference (Kerberos keytab file and ktpass command).

The following code shows the functions that are available when you enter `ktpass -?` command on the command line. This information might be different depending on the version of the toolkit that you are using.

```
C:\Program Files\Support Tools>ktpass -?
Command line options:

---------------------most useful args
[- /]          out : Keytab to produce
[- /]        princ : Principal name (user@REALM)
[- /]         pass : password to use
                     use "*" to prompt for password.
[- +]      rndPass : ... or use +rndPass to generate a random password
[- /]      minPass : minimum length for random password (def:15)
[- /]      maxPass : maximum length for random password (def:256)
---------------------less useful stuff
[- /]      mapuser : map princ (above) to this user account (default:
don't)
[- /]        mapOp : how to set the mapping attribute (default: add it)
[- /]        mapOp :  is one of:
[- /]        mapOp :        add : add value (default)
[- /]        mapOp :        set : set value
[- +]      DesOnly : Set account for des-only encryption (default:don't)
[- /]           in : Keytab to read/digest
---------------------options for key generation
[- /]       crypto : Cryptosystem to use
[- /]       crypto :  is one of:
[- /]       crypto : DES-CBC-CRC : for compatibility
[- /]       crypto : DES-CBC-MD5 : for compatibliity
[- /]       crypto : RC4-HMAC-NT : default 128-bit encryption
[- /]        ptype : principal type in question
[- /]        ptype :  is one of:
[- /]        ptype : KRB5_NT_PRINCIPAL : The general ptype-- recommended
[- /]        ptype : KRB5_NT_SRV_INST : user service instance
[- /]        ptype : KRB5_NT_SRV_HST : host service instance
[- /]         kvno : Override Key Version Number
                     Default: query DC for kvno.  Use /kvno 1 for Win2K
compat.
[- +]       Answer : +Answer answers YES to prompts.  -Answer answers
NO.
[- /]       Target : Which DC to use.  Default:detect
---------------------options for trust attributes (Windows Server 2003
Sp1 Only
[- /] MitRealmName : MIT Realm which we want to enable RC4 trust on.
[- /]  TrustEncryp : Trust Encryption to use; DES is default
[- /]  TrustEncryp :  is one of:
[- /]  TrustEncryp :        RC4 : RC4 Realm Trusts (default)
[- /]  TrustEncryp :        DES : go back to DES
```

Important: Do not use the `-pass` switch on the **ktpass** command to reset a password for a Microsoft Windows server account.

See Windows 2003 Technical Reference (Kerberos keytab file and ktpass command) for more information. You must use the -mapUser option with ktpass command to enable the KDC to create an encryption key. Otherwise, when the SPENGO token is received, it fails the validation process and the application server challenges the user for a user name and password.

Depending on the encryption type, you use the **ktpass** tool in one of the following ways to create the Kerberos keytab file. The following section shows the different types of encryption that are used by the ktpass tool. It is important that you run the ktpass -? command to determine which -crypto parameter value is expected by the particular toolkit in your Microsoft Windows environment.

- For a single DES encryption type

  From a command prompt, run the **ktpass** command:

```
ktpass -out c:\temp\myappserver.keytab
-princ HTTP/myappserver.austin.ibm.com@WSSEC.AUSTIN.IBM.COM
-mapUser myappserv
-mapOp set
-pass was1edu
-crypto DES-CBC-MD5
-pType KRB5_NT_PRINCIPAL
+DesOnly
```

Table 89. Using ktpass for a single DES encryption type.

This table describes how to use ktpass for a single DES encryption type.

Option	Explanation
`-out c:\temp\myappserver.keytab`	The key is written to this output file.
`-princ HTTP/ myappserver.austin.ibm.com@WSSEC.AUSTIN.IBM.COM`	The concatenation of the user logon name, and the realm must be in uppercase.
`-mapUser`	The key is mapped to the user, `myappserver`.
`-mapOp`	This option sets the mapping.
`-pass was1edu`	This option is the password for the user ID.
`-crypto DES-CBC-MD5`	This option uses the single DES encryption type.
`-pType KRB5_NT_PRINCIPAL`	This option specifies the `KRB5_NT_PRINCIPAL` principal value. Specify this option to avoid toolkit warning messages.
`+DesOnly`	This option generates only DES encryptions.

- For the RC4-HMAC encryption type

  Important: RC4-HMAC encryption is only supported when using a Windows 2003 Server as KDC. From a command prompt, run the **ktpass** command.

```
ktpass -out c:\temp\myappserver.keytab
-princ HTTP/myappserver.austin.ibm.com@WSSEC.AUSTIN.IBM.COM
-mapUser myappserver
-mapOp set
–pass was1edu
-crypto RC4-HMAC
-pType KRB5_NT_PRINCIPAL
```

Table 90. Using ktpass for the RC4-HMAC encryption type.

This table identifies and describes the ktpass options for RC4-HMAC encryption

Option	Explanation
`-out c:\temp\myappserver.keytab`	The key is written to this output file.
`-princ HTTP/ myappserver.austin.ibm.com@WSSEC.AUSTIN.IBM.COM`	The concatenation of the user logon name, and the realm must be in uppercase.
`-mapUser`	The key is mapped to the user, `myappserver`.
`-mapOp`	This option sets the mapping.
`-pass was1edu`	This option is the password for the user ID.
`-crypto RC4-HMAC`	This option chooses the RC4-HMAC encryption type.
`-pType KRB5_NT_PRINCIPAL`	This option specifies the `KRB5_NT_PRINCIPAL` principal value. Specify this option to avoid toolkit warning messages.

- For the RC4-HMAC-NT encryption type

  From a command prompt, run the **ktpass** command.

```
ktpass -out c:\temp\myappserver.keytab
-princ HTTP/myappserver.austin.ibm.com@WSSEC.AUSTIN.IBM.COM
-mapUser myappserver
-mapOp set
-pass was1edu
-crypto RC4-HMAC-NT
-pType KRB5_NT_PRINCIPAL
```

Table 91. Using ktpass for the RC4-HMAC encryption type. This table describes the use of ktpass for RC4-HMAC encryption types.

Option	Explanation
`-out c:\temp\myappserver.keytab`	The key is written to this output file.
`-princ HTTP/ myappserver.austin.ibm.com@WSSEC.AUSTIN.IBM.COM`	The concatenation of the user logon name, and the realm must be in uppercase.
`-mapUser`	The key is mapped to the user, `myappserver`.
`-mapOp`	This option sets the mapping.
`-pass was1edu`	This option is the password for the user ID.
`-crypto RC4-HMAC-NT`	This option chooses the RC4-HMAC-NT encryption type.
`-pType KRB5_NT_PRINCIPAL`	This option specifies the `KRB5_NT_PRINCIPAL` principal value. Specify this option to avoid toolkit warning messages.

The Kerberos keytab file is created for use with the SPNEGO TAI.

Note: A Kerberos keytab configuration file contains a list of keys that are analogous to user passwords. It is important for hosts to protect their Kerberos keytab files by storing them on the local disk, which makes them readable only be authorized users.

You make the keytab file available to WebSphere Application Server by copying the `krb5.keytab` file from the Domain Controller (LDAP machine) to the WebSphere Application Server machine.

```
ftp> bin
ftp> put c:\temp\KRB5_NT_SEV_HST\krb5.keytab
```

Results

Your active directory domain controller is properly configured to process single sign on requests to WebSphere Application Server and the SPNEGO TAI

Using the ktab command to manage the Kerberos keytab file:

The Kerberos key table manager command (Ktab) allows the product administrator to manage the Kerberos service principal names and keys stored in a local Kerberos keytab file. With the IBM Software Development Kit (SDK) or Sun Java Development Kit (JDK) 1.6 or later, you can use the ktab command to merge two Kerberos keytab files.

Kerberos service principal (SPN) name and keys listed in the Kerberos keytab file allow services running on the host to validate the incoming Kerberos or SPNEGO token request. Before configuring Kerberos or SPNEGO web authentication, the WebSphere Application Server administrator must setup a Kerberos keytab file on the host that is running WebSphere Application Server.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server Version 7.0, this function is now deprecated.

SPNEGO web authentication has taken its place to provide the following enhancements:

- Configure and enable SPNEGO Web Authentication and filters on the WebSphere Application Server side by using the administrative console.
- Provide dynamic reload of SPNEGO without having to stop and restart the WebSphere Application Server.
- Provide fallback to an application login method if the SPNEGO web authentication fails.

Important:

- It is important to protect the keytab files and make them readable only by authorized product users.
- Any updates to the Kerberos keytab file using Ktab do not affect the Kerberos database. If you change the keys in the Kerberos keytab file, you must also make the corresponding changes to the Kerberos database.

The syntax of Ktab is illustrated below by using Ktab with the `-help` operand.

```
$ ktab -help

Usage: java com.ibm.security.krb5.internal.tools.Ktab [options]
Available options:
-l                              list the keytab name and entries
-a <principal_name> [password]  add an entry to the keytab
-d <principal_name>             delete an entry from the keytab
-k <keytab_name>                specify keytab name and path with FILE: prefix
-m <source_keytab_name> <destination_keytab_name>      specify merging source keytab file name and destination keytab file name
```

Below is an example of how Ktab is used to merge the krb5Host1.keytab file to the krb5.keytab file:

```
[root@wssecjibe bin]# ./ktab -m /etc/krb5Host1.keytab /etc/krb5.keytab
Merging keytab files:   source=krb5Host1.keytab   destination=krb5.keytab
Done!
[root@wssecjibe bin]# ls /etc/krb5.*
/etc/krb5Host1.keytab/etc/krb5.keytab
/etc/krb5.keytab
```

Below is an example of how Ktab is used on a LINUX platform to add new principal names to the Kerberos keytab file, where ot56prod is the password for the Kerberos principal name:

```
[root@wssecjibe bin]# ./ktab -a
HTTP/wssecjibe.austin.ibm.com@WSSEC.AUSTIN.IBM.COM ot56prod -k /etc/krb5.keytab
Done!
Service key for principal HTTP/wssecjibe.austin.ibm.com@WSSEC.AUSTIN.IBM.COM saved
```

Below is an example of how Ktab is used on a Linux platform to list Kerberos keytab file content.

```
[root@wssecjibe bin]# ./ktab

        KVNO    Principal
        ----    ---------

        1       HTTP/wssecjibe.austin.ibm.com@WSSEC.AUSTIN.IBM.COM

[root@wssecjibe bin]# ls /etc/krb5.*
/etc/krb5.conf
/etc/krb5.keytab
```

Configuring WebSphere Application Server and enabling the SPNEGO TAI (deprecated):

Performing this task helps you, as web administrator, to ensure that WebSphere Application Server is properly configured to enable the operation of the Simple and Protected GSS-API Negotiation (SPNEGO) trust association interceptor (TAI).

Before you begin

You need to know how to use the WebSphere Application Server administrative console to manage the security configuration and have the proper authority to modify the security configuration of the application server.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application

Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

About this task

Complete the following steps to enable the operation of the SPNEGO TAI.

Procedure

1. Log on to the WebSphere Application Server administrative console.
2. Click **Security > Global security**.
3. Expand **Web security** and click **Trust association**.
4. Under the General Properties heading, select the **Enable trust association** check box, then click **Interceptors**.
5. Select the SPNEGO TAI in the list of interceptors.
6. Then click **Custom properties**.
7. Click **New** and then fill in the **Name** and **Value** fields. Click **OK**. Repeat this step for each custom property that you want to apply to the SPNEGO TAI. See “SPNEGO TAI custom properties configuration (deprecated)” on page 1469 for a complete list of SPNEGO TAI custom properties.

   Note: It is recommended that you use the **wsadmin** utility to manage the SPNEGO TAI properties. You can add, modify, and delete SPNEGO TAI properties as well as display them using wsadmin. See “Adding SPNEGO TAI properties using the wsadmin utility (deprecated)” on page 1463 to add, “Modifying SPNEGO TAI properties using the wsadmin utility (deprecated)” on page 1465 to modify, and “Deleting SPNEGO TAI properties using the wsadmin utility (deprecated)” on page 1467 to delete SPNEGO TAI properties.
8. After you finish defining your custom properties, click **Save** to store the updated SPNEGO TAI configuration.
9. Optional: If an alias for a connecting host name is added dynamically after the application server is started, you need to configure the alias. Refer to the “Using an alias host name for SPNEGO TAI or SPENGO web authentication using the administrative console (deprecated)” topic.

Results

Your SPNEGO TAI configuration is now configured for WebSphere Application Server.

Using an alias host name for SPNEGO TAI or SPENGO web authentication using the administrative console (deprecated):

When you use the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) for authentication, and you would like to use alias host name as the host name for the application server, you must configure a custom property to resolve the alias host name to the actual hostname for SPNEGO single sign-on. Then, you can dynamically add or modify an alias name in the DNS without changing the application server's configuration. If you enable this custom property you will no longer need to set alias host names through the SPNEGO configuration.

Before you begin

You must have completed the steps as described in “Creating a single sign-on for HTTP requests using the SPNEGO TAI (deprecated)” on page 1450 and “Configuring WebSphere Application Server and enabling the SPNEGO TAI (deprecated)” on page 1460 before these settings will have an effect. This configuration requires a working SPNEGO-TAI single sign-on environment.

About this task

The application server will perform a DNS lookup as an HTTP request comes in, and if the alias host name is resolved as a host name that is already configured for SPNEGO single sign-on, the application server will continue to process it. It is usually not required to add alias hostname to a SPNEGO account.

Procedure

1. Define the actual host name for the com.ibm.ws.security.spnego.SPNx.hostName variable.
   a. From administration console, click **Global security** > **Web and SIP security** > **Trust association** > **Interceptors** > **com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl** > **Custom Properties**
   b. Add or modify the com.ibm.ws.security.spnego.SPNx.hostName variable. For example:

      Name com.ibm.ws.security.spnego.SPNx.hostName

      Value *real_host_name*

      This custom property specifies the actual host name to which the application server can resolve an alias host name for SPNEGO single sign-on. You can then dynamically add or modify an alias name in the DNS without changing the configuration for the application server.

      You can optionally define the alias host name, but you are only required to define the real host name. The application server resolves the alias host name to real host name as the HTTP request is received.
2. Turn on the Canonical support flag.
   a. From administration console, click **Global security** > **Custom properties**
   b. Add or modify the com.ibm.websphere.security.krb.canonical_host variable and set it to "true".

      Name com.ibm.websphere.security.krb.canonical_host

      Value true

      This custom property specifies whether the application server uses the canonical form of the URL/HTTP host name in authenticating a client. If you set this custom property to `false`, a Kerberos ticket can contain a host name that differs from the HTTP host name header and the application server might issue the following message:

```
CWSPN0011E: An invalid SPNEGO token has been encountered while authenticating a HttpServletRequest
```

      If you set this custom property to `true`, you can avoid this error message and allow the application server to authenticate using the canonical form of the URL/HTTP host name.
3. Configure the browser. On the browser for the client machine, the alias host name needs to be configured as a trusted host.
   - For Internet Explorer:
     a. Select **Tools** > **Internet options**.
     b. Select the Security tab.
     c. Click **Local intranet** > **Sites** > **Advanced**
     d. Add the alias host name in this panel.
   - For Mozilla Firefox:
     a. Type **About:config** in the address bar and press ENTER to access configuration options.
     b. Locate the **network.negotiate-auth.trusted-uris** preference name, right-click on the preference, and select **Modify**. If you do not have this preference, right-click within the panel, and select **New** > **string**.
     c. Add alias host names in the text box, separating host names with a comma.
4. Ensure that the real host name is added to the keytab file.

   config: You can configure the keytab file in two ways:

- If com.ibm.websphere.security.krb.canonical_host is set to "true", the application server expects the real host name to be in the keytab files. Aliases are not necessary.
- If com.ibm.websphere.security.krb.canonical_host is set to false and aliases are defined, aliases need to be present in the keytab file.

Adding SPNEGO TAI properties using the wsadmin utility (deprecated):

You use the wsadmin utility to add properties for the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) in the security configuration for WebSphere Application Server.

About this task

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

Use the wsadmin utility to configure the SPNEGO TAI for WebSphere Application Server:

Procedure

1. Start WebSphere Application Server.
2. Start the command-line utility by running the **wsadmin** command from the *app_server_root*/bin directory from the Qshell command line.
3. At the **wsadmin** prompt, enter the following command:

   ```
   $AdminTask addSpnegoTAIProperties
   ```

   You can use the following parameters with this command:

Option	Description
<spnId>	This parameter is optional. It is the SPN identifier for the group of custom properties that are to be defined with this command. If you do not specify this parameter, an unused SPN identifier is assigned.
<host>	This parameter is required. It specifies the host name portion in the SPN used by the SPNEGO TAI to establish a Kerberos secure context.
<filter>	This parameter is optional. It defines the filtering criteria used by the class specified with the above attribute. If you do not specify this parameter, all HTTP requests are subject to SPNEGO authentication.
<filterClass>	This parameter is optional. It specifies the name of the Java class used by the SPNEGO TAI to select which HTTP requests will be subject to SPNEGO authentication. If you do not specify this paramter, the default filter class, com.ibm.ws.security.spnego.HTTPHeaderFilter, is used.

Option	Description
<noSpnegoPage>	This parameter is optional. It specifies the URL of a resource that contains the content the SPNEGO TAI will include in the HTTP response to be displayed by the (browser) client application if it does not support SPNEGO authentication. If you do not specify the noSpnegoPage paramter then the default is used: `"<html><head><title>SPNEGO authentication is not supported. </title></head>" + "<body>SPNEGO authentication is not supported on this client. </body></html>";`
<ntlmTokenPage>	This parameter is optional. It specifies the URL of a resource that contains the content the SPNEGO TAI will include in the HTTP response that is to be displayed by the (browser) client application when the SPNEGO token received by the interceptor (after the challenge-response handshake) contains a NT LAN manager (NTLM) token instead of the expected SPNEGO token. If you do not specify the ntlmTokenPage parameter then the default is used: `"<html><head><title>An NTLM Token was received.</title></head>" + "<body>Your browser configuration is correct, but you have not logged into a supported Windows Domain." + "<p>Please login to the application using the normal login page.</html>";`
<trimUserName>	This parameter is optional. It specifies whetheror not the SPNEGO TAI is to remove the suffix of the principal user name, starting from the "@" that precedes the Kerberos realm name. If this parameter is set to `true`, the suffix of the principal user name is removed. If this paramter is set to `false`, the suffix of the principal name is retained. The default value used is `true`.

Results

SPNEGO TAI properties have been added for this WebSphere Application Server.

Example

Example 1

The following example configures the SPNEGO TAI to intercept HTTP requests that contain `IE 6` in the user agent request header. The SPNEGO TAI uses the SPN of HTTP/ myhost.ibm.com@<default_realm> to authenticate the request originator.

```
$AdminTask addSpnegoTAIProperties -host myhost.ibm.com -filter user-agent%=IE 6
```

Example 2

The following is an example of adding SPNEGOTAIProperties for SPN1 to use the default filterClass and to intercept all requests for the host, central01.austin.ibm.com.

```
wsadmin>$AdminTask addSpnegoTAIProperties -interactive
Add SPNEGO TAI properties

Add SPNEGO TAI configuration properties.

*Host name in Service Principal Name (host): central01.austin.ibm.com
Service Principal Name identifier (spnId): 1
HTTP header filter rule (filter):
Name of class used to filter HTTP requests (filterClass):
SPNEGO not supported browser response (noSpnegoPage):
NTLM Token received browser response (ntlmTokenPage):
Trim User Name browser response (trimUserName):

Add SPNEGO TAI properties

F (Finish)
C (Cancel)

Select [F, C]: [F] f
WASX7278I: Generated command line: $AdminTask addSpnegoTAIProperties {-host central01.austin.ibm.com}
com.ibm.ws.security.spnego.SPN1.hostName=central01.austin.ibm.com
wsadmin>
```

Modifying SPNEGO TAI properties using the wsadmin utility (deprecated):

You use the wsadmin utility to modify the properties in the configuration of the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) for WebSphere Application Server.

About this task

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

You use the wsadmin utility to configure the SPNEGO TAI for WebSphere Application Server:

Procedure

1. Start WebSphere Application Server.
2. Start the command-line utility by running the **wsadmin** command from the *app_server_root*/bin directory from the Qshell command line.
3. At the **wsadmin** prompt, enter the following command:

   ```
   $AdminTask modifySpnegoTAIProperties
   ```

   You can use the following parameters with this command:

Option	Description
<spnId>	This parameter is required. It is the SPN identifier for the group of custom properties that are to be defined with this command.
<host>	This parameter is optional. It specifies the host name portion in the SPN used by the SPNEGO TAI to establish a Kerberos secure context.

Option	Description
<filter>	This parameter is optional. It defines the filtering criteria used by the class specified with the above attribute.
<filterClass>	This parameter is optional. It specifies the name of the Java class used by the SPNEGO TAI to select which HTTP requests will be subject to SPNEGO authentication. If no class is specified, all HTTP requests will be subject to SPNEGO authentication.
<noSpnegoPage>	This parameter is optional. It specifies the URL of a resource that contains the content the SPNEGO TAI will include in the HTTP response to be displayed by the (browser) client application if it does not support SPNEGO authentication. If you do not specify the noSpnegoPage attribute then the default is used: `"<html><head><title>SPNEGO authentication is not supported. </title></head>" + "<body>SPNEGO authentication is not supported on this client. </body></html>";`
<ntlmTokenPage>	This parameter is optional. The ntlmTokenPage parameter specifies the URL of a resource that contains the content the SPNEGO TAI will include in the HTTP response, which will be displayed by the (browser) client application. The (browser) client application displays this HTTP response when the browser client sends a NT LAN manager (NTLM) token instead of the expected SPNEGO token during the challange-response handshake. If you do not specify the ntlmTokenPage attribute then the default is used: `"<html><head><title>An NTLM Token was received.</title></head>" + "<body>Your browser configuration is correct, but you have not logged into a supported Windows Domain." + "<p>Please login to the application using the normal login page.</html>";`
<trimUserName>	This parameter is optional. It specifies whether (`true`) or not (`false`) the SPNEGO TAI is to remove the suffix of the principal user name, starting from the "@" that precedes the Kerberos realm name. If this attribute is set to `true`, the suffix of the principal user name is removed. If this attribute is set to `false`, the suffix of the principal name is retained. The default value used is `true`.

Results

SPNEGO TAI properties are modified for this WebSphere Application Server.

Example

Example 1

The following example configures the SPNEGO TAI to intercept HTTP requests that contain `IE 6` in

the user agent request header. The SPNEGO TAI uses the SPN of HTTP/ myhost.ibm.com@<default_realm> to authenticate the request originator. Then the example modifies the value of the filter custom property that was defined and changes it from `user-agent%=IE 6` to `host==myhost.company.com`.

```
$AdminTask addSpnegoTAIProperties -host myhost.ibm.com -filter user-agent%=IE 6
$AdminTask modifySpnegoTAIProperties -spnId 1 -filter host==myhost.company.com
```

Example 2

This is an example of modifying the SPNEGO TAI for SPN1 properties to add a filter for host central01.austin.ibm.com.

```
wsadmin>$AdminTask modifySpnegoTAIProperties -interactive
Modify SPNEGO TAI properties

Modify SPNEGO TAI configuration properties

*Service Principal Name identifier (spnId): 1
Host name in Service Principal Name (host): central01.austin.ibm.com
HTTP header filter rule (filter): request-url!=noSPNEGO;request-url%=snoop
Name of class used to filter HTTP requests (filterClass):
SPNEGO not supported browser response (noSpnegoPage):
NTLM Token received browser response (ntlmTokenPage):
Trim User Name browser response (trimUserName):

Modify SPNEGO TAI properties

F (Finish)
C (Cancel)

Select [F, C]: [F] f
WASX7278I: Generated command line: $AdminTask modifySpnegoTAIProperties {-spnId
1 -host w2003secdev.austin.ibm.com -filter request-url!=noSPNEGO;request-url%=sn
oop}
com.ibm.ws.security.spnego.SPN1.filter=request-url!=noSPNEGO;request-url%=snoop
com.ibm.ws.security.spnego.SPN1.hostName=central01.austin.ibm.com
wsadmin>
```

Deleting SPNEGO TAI properties using the wsadmin utility (deprecated):

You use the wsadmin utility to delete properties in the configuration of the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) for WebSphere Application Server.

About this task

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

You use the wsadmin utility to configure the SPNEGO TAI for WebSphere Application Server:

Procedure

1. Start WebSphere Application Server.
2. Start the command-line utility by running the **wsadmin** command from the *app_server_root*/bin directory from the Qshell command line.

3. At the **wsadmin** prompt, enter the following command:
   ```
   $AdminTask deleteSpnegoTAIProperties
   ```

   You can use the following parameters with this command:

Option	Description
<spnId>	This is an optional parameter. It is the SPN identifier for the group of custom properties that are to be deleted with this command. If you do not specify this parameter, all SPNEGO TAI custom properties are deleted.

Results

SPNEGO TAI properties are deleted for this WebSphere Application Server.

Example

Example 1

The following example deletes all the SPNEGO TAI properties for SPN2

```
wsadmin>$AdminTask deleteSpnegoTAIProperties {-spnId 2}
```

Example 2

The following example deletes all SPNEGO TAI properties

```
wsadmin>$AdminTask deleteSpnegoTAIProperties
com.ibm.ws.security.spnego.SPN1.filter=request-url!=noSPNEGO;request-url%=snoop
com.ibm.ws.security.spnego.SPN1.hostName=central01.austin.ibm.com
com.ibm.ws.security.spnego.SPN2.hostName=wssecpd.austin.ibm.com
wsadmin>
```

Displaying SPNEGO TAI properties using the wsadmin utility (deprecated):

You use the wsadmin utility to display the properties in the configuration of the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) for WebSphere Application Server.

About this task

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

You use the wsadmin utility to configure the SPNEGO TAI for WebSphere Application Server:

Procedure

1. Start WebSphere Application Server.
2. Start the command-line utility by running the **wsadmin** command from the *app_server_root*/bin directory from the Qshell command line.
3. At the **wsadmin** prompt, enter the following command:
   ```
   $AdminTask showSpnegoTAIProperties
   ```

   You can use the following parameters with this command:

Option	Description
<spnId>	This is an optional parameter. It is the service principal name (SPN) identifier for the group of custom properties that are to be displayed with this command. If you do not specify this parameter, all SPNEGO TAI custom properties are displayed.

Results

SPNEGO TAI properties are displayed for this WebSphere Application Server.

Example

Example 1

The following example displays all SPNEGO TAI properties.

```
wsadmin>$AdminTask showSpnegoTAIProperties
com.ibm.ws.security.spnego.SPN1.filter=request-url!=noSPNEGO;request-url%=snoop
com.ibm.ws.security.spnego.SPN1.hostName=central01.austin.ibm.com
com.ibm.ws.security.spnego.SPN2.hostName=wssecpd.austin.ibm.com
wsadmin>
```

Example 2

The following example displays SPNEGO TAI properties for SPN1 and host, central01.austin.ibm.com.

```
wsadmin>$AdminTask showSpnegoTAIProperties -interactive
Show SPNEGO TAI configuration properties.

Display SPNEGO TAI configuration properties.

Service Principal Name identifier (spnId): 1

Show SPNEGO TAI configuration properties.

F (Finish)
C (Cancel)

Select [F, C]: [F]
WASX7278I: Generated command line: $AdminTask showSpnegoTAIProperties {-spnId 1}

com.ibm.ws.security.spnego.SPN1.filter=request-url!=noSPNEGO;request-url%=snoop
com.ibm.ws.security.spnego.SPN1.hostName=central01.austin.ibm.com
com.ibm.ws.security.spnego.SPN1.trimUserName=true
wsadmin>
```

SPNEGO TAI custom properties configuration (deprecated):

The Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) custom configuration properties control different operational aspects of the SPNEGO TAI. You can specify different property values for each application server.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

Each of the properties defined in the following table is specified in the Custom Properties panel for the SPNEGO TAI using the administrative console facility. For convenience, you can optionally place these properties in a properties file. In this case, the SPNEGO TAI loads the configuration properties from the file instead of the Custom Properties panel definition. Refer to com.ibm.ws.security.spnego.propertyReloadFile property as defined in “SPNEGO TAI JVM configuration custom properties (deprecated)” on page 1477.

To assign unique property names that identify each possible SPN, an SPN<id> is embedded in the property name and used to group the properties that are associated with each SPN. The SPN<id>s are numbered sequentially for each property group.

Table 92. SPNEGO TAI custom properties.

This table lists the SPNEGO TAI custom properties.

Property Name	Required	Default Value
“com.ibm.ws.security.spnego.SPN<id>.enableCredDelegate”	No	false
“com.ibm.ws.security.spnego.SPN<id>.filter”	No	See the description that follows.
“com.ibm.ws.security.spnego.SPN<id>.filterClass” on page 1472	No	See the description that follows.
“com.ibm.ws.security.spnego.SPN<id>.hostName” on page 1472	Yes	None
“com.ibm.ws.security.spnego.SPN<id>.NTLMTokenReceivedPage” on page 1472	No	See the description that follows.
“com.ibm.ws.security.spnego.SPN<id>.spnegoNotSupportedPage” on page 1472	No	See the description that follows.
“com.ibm.ws.security.spnego.SPN<id>.trimUserName” on page 1472	No	true

Note: The following commands tasks can be used to operate on these SPNEGO TAI properties:

- “Adding SPNEGO TAI properties using the wsadmin utility (deprecated)” on page 1463
- “Deleting SPNEGO TAI properties using the wsadmin utility (deprecated)” on page 1467
- “Modifying SPNEGO TAI properties using the wsadmin utility (deprecated)” on page 1465
- “Displaying SPNEGO TAI properties using the wsadmin utility (deprecated)” on page 1468

com.ibm.ws.security.spnego.SPN<id>.enableCredDelegate:

This property is optional. It indicates whether or not the Kerberos delegated credentials are stored by the SPNEGO TAI. This property enables the capability for an application to retrieve the stored credentials and propagate them to other applications downstream for additional SPNEGO authentication.

This property requires use of the advanced Kerberos credential delegation feature and requires development of custom logic by the application developer. The developer must interact directly with the Kerberos Ticket Granting Service (TGS) to obtain a Ticket Granting Ticket (TGT) using the delegated Kerberos credentials on behalf of the end-user who originated the request. The developer must also construct the appropriate Kerberos SPNEGO token and include it in the HTTP request to continue the downstream SPNEGO authentication process, including handling additional SPNEGO challenge-response exchange, if necessary.

com.ibm.ws.security.spnego.SPN<id>.filter:

This property is optional. It defines the filtering criteria that is used by the specified class with the com.ibm.ws.security.spnego.SPN<id>.filterClass property. It defines arbitrary criteria that is meaningful to the implementation class used.

The com.ibm.ws.security.spnego.HTTPHeaderFilter default implementation class uses this property to define a list of selection rules that represent conditions that are matched against the HTTP request headers to determine whether or not the HTTP request is selected for SPNEGO authentication.

Each condition is specified with a key-value pair, separated from each other by a semicolon. The conditions are evaluated from left to right, as they display in the specified property. If all conditions are met, the HTTP request is selected for SPNEGO authentication.

The key and value in the key-value pair are separated by an operator that defines which condition is checked. The key identifies an HTTP request header to extract from the request and its value is compared with the value that is specified in the key-value pair according to the operator specification. If the header that is identified by the key is not present in the HTTP request, the condition is treated as not being met.

Any of the standard HTTP request headers can be used as the key in the key-value pairs. Refer to the HTTP specification for the list of valid headers. In addition, two keys are defined to extract information from the request, also useful as a selection criterion, which is not available through standard HTTP request headers. The remote-address key is used as a pseudo header to retrieve the remote TCP/IP address of the client application that sent the HTTP request. The request-URL key is used as a pseudo header to retrieve the URL that is used by the client application to make the request. The interceptor uses the result of the getRequestURL operation in the javax.servlet.http.HttpServletRequest interface to construct the web address. If a query string is present, the result of the getQueryString operation in the same interface is also used. In this case, the complete URL is constructed as follows:

```
String url = request.getRequestURL() + '?' + request.getQueryString();
```

The following operators and conditions are defined:

Table 93. Filter conditions and operations.

This table defines the conditions and operators used in filtering and gives examples.

Condition	Operator	Example
Match exactly	= = Arguments are compared as equal.	host=host.my.company.com
Match partially (includes)	%= Arguments are compared with a partial match being valid.	user-agent%=IE 6
Match partially (includes one of many)	^= Arguments are compared with a partial match being valid for one of many arguments specified.	request-url^=webApp1|webApp2|webApp3
Does not match	!= Arguments are compared as not equal.	request-url!=noSPNEGO
Greater than	> Arguments are compared lexogaphically as greater than.	remote-address>192.168.255.130
Less than	< Arguments are compared lexographically as less than.	remote-address<192.168.255.135

com.ibm.ws.security.spnego.SPN<id>.filterClass:

This property is optional. It specifies the name of the Java class that is used by the SPNEGO TAI to select which HTTP requests are subject to SPNEGO authentication.

If no class is specified, the default com.ibm.ws.security.spnego.HTTPHeaderFilter implementation class is used. The Java class that is specified must implement the `com.ibm.wsspi.security.spnego.SpnegoFilter` interface. A default implementation of this interface is provided. Specify the `com.ibm.ws.security.spnego.HTTPHeaderFilter` class to use the default implementation. This class uses the selection rules specified with the `com.ibm.ws.security.spnego.SPN<id>.filter` property.

com.ibm.ws.security.spnego.SPN<id>.hostName:

This property is required. It specifies the hostname in the SPN used by the SPNEGO TAI to establish a Kerberos secure context. It has no default value.

Note: The hostname is the long form of hostname. For example, `myHostName.austin.ibm.com`.
The Kerberos SPN is a string of the form `HTTP/hostname@realm`. The complete SPN is used with the Java Generic Security Service (JGSS) by the SPNEGO provider to obtain the security credential and security context that are used in the authentication process.

com.ibm.ws.security.spnego.SPN<id>.NTLMTokenReceivedPage:

This property is optional. It specifies the web address of a resource that contains the content that the SPNEGO TAI includes in the HTTP response that the (browser) client application displays when the SPNEGO token is received by the interceptor when the challenge-response handshake contains a NT LAN Manager (NTLM) token instead of the expected SPNEGO token.

It can specify a web (http://) or a file (file://) resource. If this property is not specified or the interceptor cannot find the specified resource, the following content is used:

```
<html><head><title>An NTLM Token was received.</title></head>
<body>Your browser configuration is correct, but you have not logged into a supported
Microsoft(R) Windows(R) Domain.
<p>Please login to the application using the normal login page.</html>
```

com.ibm.ws.security.spnego.SPN<id>.spnegoNotSupportedPage:

This property is optional. It specifies the web address of a resource that contains the content that the SPNEGO TAI includes in the HTTP response that the (browser) client application displays if it does not support SPNEGO authentication. It can specify a Web (http://) or a file (file://) resource.

If this property is not specified or the interceptor cannot find the specified resource, the following content is used:

```
<html><head><title>SPNEGO authentication is not supported</title></head>
<body>SPNEGO authentication is not supported on this client</body></html>;
```

com.ibm.ws.security.spnego.SPN<id>.trimUserName:

This property is optional. It specifies whether (`true`) or not (`false`) the SPNEGO TAI is to remove the suffix of the principal user name, starting from the "@" that precedes the Kerberos realm name.

If this property is set to `true`, the suffix of the principal user name is removed. If this property is set to `false`, the suffix of the principal name is retained. The default value used is `true`. For example,

When com.ibm.ws.security.spnego.SPN<id>.trimUserName = true

```
bobsmith@myKerberosRealm  becomes  bobsmith
```

When com.ibm.ws.security.spnego.SPN<id>.trimUserName = false

```
bobsmith@myKerberosRealm  remains  bobsmith@myKerberosRealm
```

SPNEGO TAI configuration requirements (deprecated):

The configuration that is used by the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) on each selected application server is governed by various system requirements.

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, the SPNEGO TAI was deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

The following list of configuration requirements highlights those attributes, properties, qualities, restrictions, exclusions, inclusions, and dependencies that you need to be aware of when planning a WebSphere Application Server configuration that incorporates the use of the SPNEGO TAI.

Table 94. SPNEGO TAI requirements.

This table lists the SPNEGO TAI configuration requirements.

Function item	Description
SPNEGO TAI	The SPNEGO TAI is a server side solution in WebSphere Application Server. Client-side applications are responsible for generating the SPNEGO token for use by the SPNEGO TAI.
Microsoft Windows	Microsoft Windows Servers with Active Directory domain and its associated Kerberos key distribution center (KDC) is required. For information on the supported Microsoft Windows Servers, see the System Requirements for WebSphere Application Server Version 8.0 on Windows.
Client application (browser or .NET client)	A browser (client application) or .NET client that supports the SPNEGO authentication mechanism, as defined in IETF RFC 2478 is required.
Simple and Protected GSS-API Negotiation Mechanism (SPNEGO)	SPNEGO authentication, as defined in IETF RFC 2478 is used.
Internet browsers	• Use Microsoft Internet Explorer version 5.5 or higher • Use Mozilla Firefox version 1.0
Kerberos Level	Kerberos version 5 is required.
WebSphere Application Server	Version 7.0 is required.
Java SDK level	Java 6.0 SDK is required.
Encryption Types	RC4-HMAC encryption is only supported when using a Windows 2003 Server as Kerberos key distribution center (KDC).
J2EE client	Client application (browser or .NET client) A browser (client application) or .NET client that supports the SPNEGO authentication mechanism, as defined in IETF RFC 2478 is required.

Configuring the client browser to use SPNEGO TAI (deprecated):

You can configure your browser to utilize the Simple and Protected GSS-API Negotiation (SPNEGO) mechanism. Authentication of your browser requests are processed by the SPNEGO trust association interceptor (TAI) in the WebSphere Application Server.

Before you begin

You need to know how to display and set options in the Microsoft Internet Explorer browser or any other browser (such as Firefox). You must have a browser installed that supports SPNEGO authentication.

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

About this task

Complete the following steps to ensure that your Microsoft Internet Explorer browser is enabled to perform SPNEGO authentication.

Procedure

1. At the desktop, log in to the windows active directory domain.
2. Activate Internet Explorer.
3. In the Internet Explorer window, click **Tools > Internet Options > Security** tab.
4. Select the **Local intranet** icon and click **Sites**.
5. In the Local intranet window, ensure that the "check box" to include all local (intranet) not listed in other zones is selected, then click **Advanced**.
6. In the **Local intranet** window, fill in the Add this web site to the zone field with the web address of the host name so that the single sign-on (SSO) can be enabled for the list of websites shown in the websites field. Your site information technology staff provides this information. Click **OK** to complete this step and close the Local intranet window.
7. On the **Internet Options** window, click the **Advanced** tab and scroll to **Security settings**. Ensure that the **Enable Integrated Windows Authentication (requires restart)** box is selected.
8. Click **OK**. Restart your Microsoft Internet Explorer to activate this configuration.

Results

Complete the following steps to ensure that your Firefox browser is enabled to perform SPNEGO authentication.

1. At the desktop, log in to the windows active directory domain.
2. Activate Firefox.
3. At the address field, type **about:config**.
4. In the Filter, type **network.n**
5. Double click on **network.negotiate-auth.trusted-uris**. This preference lists the sites that are permitted to engage in SPNEGO Authentication with the browser. Enter a comma-delimited list of trusted domains or URLs.

   Note: You must set the value for **network.negotiate-auth.trusted-uris**.

6. If the deployed SPNEGO solution is using the advanced Kerberos feature of Credential Delegation double click on **network.negotiate-auth.delegation-uris**. This preference lists the sites for which the browser may delegate user authorization to the server. Enter a comma-delimited list of trusted domains or URLs.
7. Click **OK**. The configuration appears as updated.
8. Restart your Firefox browser to activate this configuration.

Your Internet browser is properly configured for SPNEGO authentication. You can use applications that are deployed in WebSphere Application Server that use secured resources without being repeatedly requested for an ID and password.

Configuring JVM custom properties, filtering HTTP requests, and enabling SPNEGO TAI in WebSphere Application Server (deprecated):

Performing this task helps you, as web administrator, to ensure that WebSphere Application Server is configured to enable the operation of the Simple and Protected GSS-API Negotiation mechanism (SPNEGO) trust association interceptor (TAI) with the required Java virtual machine (JVM) property and with the appropriate filtering of HTTP requests.

Before you begin

You need to know how to use the WebSphere Application Server administrative console to manage the security configuration and have the proper authority to modify the security configuration of the application server.

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

About this task

Verify the configuration of your SPNEGO TAI. The deployment of the SPNEGO TAI can vary from a single WebSphere Application Server system on which a single application is running to a large multinode WebSphere Application Server, Network Deployment (ND) cell, with dozens of application servers, hosting many applications. Every SPNEGO TAI is installed at the cell level. You must be aware of your particular SPNEGO TAI configuration.

The default behavior of the SPNEGO TAI is to not intercept HTTP requests. This default behavior ensures that the SPNEGO TAI can be installed into an existing cell, configured for a single application server and not change any other application servers in the cell. Other WebSphere Application Servers can run exactly as before within a given configuration.

Decide whether or not to use the sample SPN<id>.filterClass and determine the exact filter properties to use.

Note: The default behavior of the SPNEGO TAI is to use the `com.ibm.ws.security.spnego.SPN<id>.filterClass` and intercept all requests.
If the default behavior is not appropriate, you can use a customer provided class, or extend or modify the sample class as required. The system programmer interface, `com.ibm.ws.security.spnego.SpnegoFilter`

allows you to implement a custom filter to determine whether or not to intercept a particular HTTP request. With the default implementation, you can set filter rules for coarse as well as fine-grained criteria in selecting which HTTP requests to intercept.

Note: For an alternative to the steps below for enabling the SPNEGO TAI, you can use scripting to perform the operation. See "Enabling the SPNEGO TAI as JVM custom property using scripting (deprecated)" for the details.

Complete the following steps to enable the operation of the SPNEGO TAI with your selected filtering and with the JVM required property.

Procedure

1. Log on to WebSphere Application Server administrative console.
2. Click **Servers > Application servers**.
3. Select the appropriate server. Under Server Infrastructure, expand **Java and process management > Process Definition**.
4. Click **Java virtual machine**. Under Additional Properties, click **Custom Properties**. Create a new custom property, if required, by clicking **New**, then code `com.ibm.ws.security.spnego.isEnabled` in the name field and `true` in the value field.
5. Click **Apply > OK** to save the configuration
6. Identify when the SPNEGO TAI intercepts a given request. A set of filter properties is provided, but you must determine what is appropriate and modify the `com.ibm.ws.security.spnego.SPN<id>.filterClass` accordingly.

Results

The application server is configured and ready to provide a single sign-on environment for end users who have successfully authenticated in a Microsoft Active Directory domain. You must restart each application server that is configured for SPNEGO web authentication. Then your SPNEGO TAI is set to filter HTTP request when it is operating.

Enabling the SPNEGO TAI as JVM custom property using scripting (deprecated):

You use the wsadmin utility to enable the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) for WebSphere Application Server.

Before you begin

Before starting this task, the wsadmin tool must be running. See the information about starting the wsadmin scripting client using wsadmin scripting.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

About this task

Perform the following steps to enable the SPNEGO TAI:

Procedure

1. Identify the server and assign it to the `server1` variable:
   - Using Jacl:

     ```
     set server1 [$AdminConfig getid /Cell:mycell/Node:mynode/Server:server1/]
     ```

   - Using Jython:

     ```
     server1 = AdminConfig.getid("/Cell:mycell/Node:mynode/Server:server1/")
     print server1
     ```

   Example output:

   ```
   server1(cells/mycell/nodes/mynode|servers/seerver1|server.xml#Server_1)
   ```

2. Identify the Java virtual machine (JVM) belonging to this server and assign it to the `jvm` variable:
   - Using Jacl:

     ```
     set jvm [$AdminConfig list JavaVirtualMachine $server1]
     ```

   - Using Jython:

     ```
     jvm = AdminConfig.list('JavaVirtualMachine',server1)
     ```

   Example output:

   ```
   (cells/mycell/nodes/mynode/servers/server1:server.xml#JavaVirtualMachine_1)
   (cells/mycell/nodes/mynode/servers/server1:server.xml#JavaVirtualMachine_2)
   ```

3. Identify the controller JVM of the server:
   - Using Jacl:

     ```
     set cjvm [lindex $jvm 0]
     ```

   - Using Jython:

     ```
     # get line separator
     import java
     lineSeparator = java.lang.System.getProperty('line.separator')
     arrayJVMs = jvm.split(lineSeparator)
     cjvm = arrayJVMs[0]
     ```

4. Modify the generic JVM arguments to enable SPNEGO TAI:
   - Using Jacl:

     ```
     set attr_name           [list name com.ibm.ws.security.spnego.isEnabled]
     set attr_value          [list value true]
     set attr_required       [list required false]
     set attr_description    [list description "Enabled SPNEGO TAI"]

     set attrs [list $attr_name $attr_value $attr_required $attr_description]

     $AdminConfig create Property $cjvm $attrs
     ```

   - Using Jython:

     ```
     attr_name  = ['name', "com.ibm.ws.security.spnego.isEnabled"]
     attr_value = ['value', "true"]
     attr_required = ['required', "false"]
     attr_description = ['description', "Enabled SPNEGO TAI"]
     attr_list = [attr_name, attr_value, attr_required, attr_description]
     property=['systemProperties',[attr_list]]
     AdminConfig.modify(cjvm, [property])
     ```

5. Save the configuration changes.

SPNEGO TAI JVM configuration custom properties (deprecated):

Java virtual machine (JVM) custom properties control the operation of the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI).

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

The following JVM custom properties control operation of the SPNEGO TAI. Different custom property values can be specified for each application server.

Table 95. JVM configuration custom properties.

This table lists the SPNEGO JVM configuration custom properties.

Custom Property Name	Required	Value Type	Default Value	Recommended Value
com.ibm.ws.security.spnego.isEnabled	No	Boolean	False	True

com.ibm.ws.security.spnego.propertyReloadFile	No	String	None	For Windows
				`C:\temp\TAI.props`
				
				For UNIX
				`/tmp/TestTAI.Properties`

com.ibm.ws.security.spnego.propertyReloadTimeout	No	Integer	None	120

com.ibm.ws.security.spnego.isEnabled
Use this custom property to enable or disable operation of the SPNEGO TAI in a given application server. When set to `false`, the SPNEGO TAI is disabled and not used by the web authentication module for authenticating any web requests. When set to `true`, the SPNEGO TAI is enabled and used by the web authentication module for authenticating any web requests.

com.ibm.ws.security.spnego.propertyReloadFile
Use this custom property to identify the file that contains configuration properties for the SPNEGO TAI, when it is not convenient to stop and restart the application server. The properties contained in this file can be reloaded to configure the SPNEGO TAI.

Important: The properties that are defined in the specified file override any properties defined using the administrative console.

A sample of this reload file follows:

```
#########################################################
# Template properties files for SPNEGO TAI
#
# Where possible defaults have been provided.
#
#########################################################

#---------------------------------------------------------
# Hostname
#---------------------------------------------------------
#com.ibm.ws.spnego.SPN1.HostName=wsecurity.austin.ibm.com

#---------------------------------------------------------
# (Optional) SpnegoNotSupportedPage
#---------------------------------------------------------
#com.ibm.ws.spnego.SPN1.SpnegoNotSupportedPage=

#---------------------------------------------------------
# (Optional) NTLMTokenReceivedPage
#---------------------------------------------------------
#com.ibm.ws.spnego.SPN1.NTLMTokenReceivedPage=
```

```
#-----------------------------------------------------------
# (Optional) FilterClass
#-----------------------------------------------------------
#com.ibm.ws.spnego.SPN1.FilterClass=com.ibm.ws.spnego.HTTPHeaderFilter

#-----------------------------------------------------------
# (Optional) Filter
#-----------------------------------------------------------
#com.ibm.ws.spnego.SPN1.Filter=
```

Important: If com.ibm.ws.security.spnego.propertyReloadFile custom property is set, but the com.ibm.ws.security.spnego.propertyReloadTimeout custom property is not, then the SPNEGO TAI is not initialized.

com.ibm.ws.security.spnego.propertyReloadTimeout
Use this custom property to specify a time interval in seconds that elapses after which the SPNEGO TAI reloads the configuration properties. Also, the SPNEGO TAI reloads the configuration properties if the file that is identified by the com.ibm.ws.security.spnego.propertyReloadFile custom property changed since the last time the configuration custom properties were retrieved. This time interval in seconds must be specified as a positive integer.

Important:

- If the com.ibm.ws.security.spnego.propertyReloadFile custom property and the com.ibm.ws.security.spnego.propertyReloadTimeout custom property are not set, then the SPNEGO TAI properties are only loaded once from the SPNEGO TAI custom properties defined in the WebSphere Application Server configuration data. This one time loading occurs when the JVM is initialized.
- If com.ibm.ws.security.spnego.propertyReloadTimeout custom property is set, but the com.ibm.ws.security.spnego.propertyReloadFile custom property is not, then the SPNEGO TAI is not initialized. “Configuring JVM custom properties, filtering HTTP requests, and enabling SPNEGO TAI in WebSphere Application Server (deprecated)” on page 1475 or how to configure the JVM custom properties for SPNEGO TAI.

Remember: You can also use the wsadmin command for the AdminConfig scripting object to interactively set the com.ibm.ws.security.spnego.isEnabled custom property. See “Enabling the SPNEGO TAI as JVM custom property using scripting (deprecated)” on page 1476 for more information.

The following custom properties are not used directly by the SPNEGO TAI; however, they affect the operation of the core security runtime and can also be used for problem determination.

Table 96. JVM configuration custom properties.

This table describes the JVM configuration custom properties

Custom Property Name	Required	Value Type	Default Value	Recommended Value
com.ibm.security.jgss.debug	No	String	None	"off" or "all"
com.ibm.security.krb5.Krb5Debug	No	String	None	"off" or "all"
java.security.properties	No	String	None	
javax.security.auth.useSubjectCredsOnly	Yes	Boolean	True	False

com.ibm.security.jgss.debug
This custom property is optional. It can be used to collect diagnostic trace information for problem determination in the Java Generic Security Service (JGSS) application programmer interface (API) implementation. The value can be set to `all` or `off` to enable or disable tracing, respectively. See Java Generic Security Service User's Guide for specific JGSS API information.

com.ibm.security.krb5.Krb5Debug
This custom property is optional. It can be used to collect additional diagnostic trace information for problem determination in the JGSS implementation. The value can be set to `all` or `off` to enable or disable tracing, respectively.

java.security.properties
This property is optional. It can be used when different application servers in a cell have different security requirements and it is not convenient to modify the global `java.security` file for the entire cell. In such situations, the java.security.properties custom property is used to specify the location of the `java.security` file used by the JVM for each application server.

javax.security.auth.useSubjectCredsOnly
JGSS includes an optional Java Authentication and Authorization Service (JAAS) login facility that saves `Principal` credentials and secret keys in the `Subject` of the application's JAAS login context. JGSS retrieves credentials and secret keys from the `Subject` by default. This feature can be disabled by setting the Java property javax.security.auth.useSubjectCredsOnly to `false`.

Attention: The SPNEGO TAI does not use the optional JAAS login module. The javax.security.auth.useSubjectCredsOnly property must be set to `false`.

Mapping Kerberos client principal name to WebSphere user registry ID for SPNEGO TAI (deprecated):

You can use a system programming interface to customize the behavior of the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) by implementing arbitrary mappings of the end-user's identity, which is retrieved from Microsoft Active Directory to the identity that is used in the WebSphere Application Server security registry.

Before you begin

You need to perform some administrative tasks in the WebSphere Application Server environment to use SPNEGO TAI and to ensure that the requester's identity matches the identity in the WebSphere Application Server user registry.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

Note: Make sure the following tasks have been performed successfully:

1. Configuring the web browser to use SPNEGO. See “Configuring the client browser to use SPNEGO TAI (deprecated)” on page 1473
2. Configuring Java virtual machine (JVM) properties, custom SPNEGO TAI properties, and enabling the SPNEGO TAI. See “Configuring JVM custom properties, filtering HTTP requests, and enabling SPNEGO TAI in WebSphere Application Server (deprecated)” on page 1475

About this task

In the simplest deployment of the SPNEGO TAI, it is assumed that the requester's identity in the WebSphere Application Server user registry is identical to the identity retrieved. This is the case when Microsoft Windows Active Directory server is the lightweight directory access protocol (LDAP) server used in WebSphere Application Server. This is default behavior of the SPNEGO TAI.

You do not need to use this simple deployment of the SPNEGO TAI. WebSphere Application Server can use a different registry, such as a local OS, LDAP, or custom registry instead of the Microsoft Active Directory. If WebSphere Application Server uses a different registry than the Microsoft Active Directory, then a mapping from the Microsoft Windows user Id to a WebSphere Application Server user Id is necessary.

Procedure

Use the JAAS custom login module to perform any custom mapping of a client Kerberos principal name from the Microsoft Active Directory to the WebSphere user registry identity. The JAAS custom login module is a plug-in mechanism that is defined for authenticating incoming and outgoing requests in WebSphere Application Server and is inserted before the `ltpaLoginModule`. The JAAS custom login module retrieves a client Kerberos principal name in the `javax.security.auth.Subject` using `subject.getPrincipals(KerberosPrincipal.class)` method, maps the client Kerberos principal name to the WebSphere user registry identity, and inserts the mapping identity in the hash table property `com.ibm.wsspi.security.cred.userId`. The `ltpaLoginModule` then uses the mapped identity to create a `WSCredential`.

Note: The custom login module can also supply the full set of security properties in the `javax.security.auth.Subject` in the `com.ibm.wsspi.security.tai.TAIResult` to fully assert the mapped identity. When the identity is fully asserted, the `wsMapDefaultInboundLoginModule` maps those security properties to a `WSCredential`.

A sample of the custom login module follows:

```
package com.ibm.ws.security.server.lm;

import java.util.Map;
import java.lang.reflect.Array;
import javax.security.auth.Subject;
import javax.security.auth.callback.*;
import javax.security.auth.login.LoginException;
import javax.security.auth.spi.LoginModule;
import javax.security.auth.kerberos.*;
import com.ibm.websphere.security.auth.WSLoginFailedException;
import com.ibm.wsspi.security.token.AttributeNameConstants;

/**
 *
 * @author IBM Corporation
 * @version 1.0
 * @since 1.0
 *
 */

public class sampleSpnegoMappingLoginModule implements LoginModule {
    /*
     *
     * Constant that represents the name of this mapping module.  Whenever this sample
     * code is used to create a class with a different name, this value should be changed.
     *
     */
    private final static String MAPPING_MODULE_NAME = "com.ibm.websphere.security.sampleSpnegoMappingLoginModule";

    private String mapUid = null;
    /**
     * Construct an uninitialized WSLoginModuleImpl object.
     */
    public sampleSpnegoMappingLoginModule() {
        debugOut("sampleSpnegoMappingLoginModule() entry");
        debugOut("sampleSpnegoMappingLoginModule() exit");
    }

    /**
```

```
 * Initialize this login module.
 *
 *
 * This is called by the  LoginContext after this login module is
 * instantiated. The relevant information is passed from the LoginContext
 * to this login module. If the login module does not understands any of the data
 * stored in the sharedState and options parameters,
 * they can be ignored.
 *
 *
 * @param subject The subject to be authenticated.
 * @param callbackHandler
 *                A  CallbackHandler for communicating with the end user to gather
 *                login information (e.g., username and password).
 * @param sharedState
 *                The state shared with other configured login modules.
 * @param options The options specified in the login configuration for this particular login module.
 */
public void initialize(Subject subject, CallbackHandler callbackHandler,
                       Map sharedState, Map options) {
    debugOut("initialize(subject = \"" + subject.toString() +
             "\", callbackHandler = \"" + callbackHandler.toString() +
             "\", sharedState = \"" + sharedState.toString() +
             "\", options = \"" + options.toString() + "\")");

    this.subject = subject;
    this.callbackHandler = callbackHandler;
    this.sharedState = sharedState;
    this.options = options;

    debug = "true".equalsIgnoreCase((String)this.options.get("debug"));

    debugOut("initialize() exit");
}

/**
 *
 * Method to authenticate a Subject (phase 1).
 *
 *
 *
 * This method authenticates a Subject. It uses CallbackHandler to gather
 * the Subject information, like username and password for example, and verify these
 * information. The result of the authentication is saved in the private state within
 * this login module.
 *
 *
 * @return  true if the authentication succeeded, or false
 *         if this login module should be ignored.
 * @exception LoginException
 *                   If the authentication fails.
 */
public boolean login() throws LoginException
{
    debugOut("sampleSpnegoMappingLoginModule.login() entry");

    boolean succeeded = false;
    java.util.Set krb5Principals= subject.getPrincipals(KerberosPrincipal.class);
    java.util.Iterator krb5PrincIter = krb5Principals.iterator();

    while (krb5PrincIter.hasNext()) {
        Object princObj = krb5PrincIter.next();
        debugOut("Kerberos principal name: "+ princObj.toString());

        if (princObj != null && princObj.toString().equals("utle@WSSEC.AUSTIN.IBM.COM")){
            mapUid = "user1";
            debugOut("mapUid: "+mapUid);
```

```
                            java.util.Hashtable customProperties = (java.util.Hashtable)
                            sharedState.get(AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY);
                            if (customProperties == null) {
                                customProperties = new java.util.Hashtable();
                            }
                            succeeded = true;
                            customProperties.put(AttributeNameConstants.WSCREDENTIAL_USERID, mapUid);

                            Map<String,java.util.Hashtable)>
                            mySharedState=(Map<String,java.util.Hashtable>)sharedState;
                            mySharedState.put((AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY.customProperties)

                            debugOut("Add a mapping user ID to Hashtable, mapping ID = "+mapUid);

                           }
                            debugOut("login() custom properties = " + customProperties);
        }
    }

    succeeded = true;
    debugOut("sampleSpnegoMappingLoginModule.login() exit");

    return succeeded;
}

/**
 *
 * Method to commit the authentication result (phase 2).
 *
 *
 *
 * This method is called if the LoginContext's overall authentication
 * succeeded (the revelant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL login module
 * succeeded).
 *
 *
 * @return true if the commit succeeded, or false
 *         if this login module should be ignored.
 * @exception LoginException
 *                   If the commit fails.
 */
public boolean commit() throws LoginException
{
    debugOut("commit()");

    debugOut("commit()");

    return true;
}

/**
 * Method to abort the authentication process (phase 2).
 *
 *
 * This method is called if the LoginContext's overall authentication
 * failed (the revelant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL login module
 * did not succeed).
 *
 *
 *
 * If this login module's authentication attempt succeeded, then this method cleans
 * up the previous state saved in phase 1.
 *
 *
 * @return true if the abort succeeded, or false
 *         if this login module should be ignored.
```

```
     * @exception LoginException
     *                   If the abort fails.
     */
    public boolean abort() throws LoginException {
        debugOut("abort() entry");
        debugOut("abort() exit");
        return true;
    }

    /**
     * Method which logs out a Subject.
     *
     * @return true if the logout succeeded, or false
     *         if this login module should be ignored.
     * @exception LoginException
     *                   If the logout fails.
     */
    public boolean logout() throws LoginException
    {
        debugOut("logout() entry");
        debugOut("logout() exit");

        return true;
    }

    private void cleanup()
    {
        debugOut("cleanup() entry");
        debugOut("cleanup() exit");
    }

    /*
     *
     * Private method to print trace information.  This implementation uses System.out
     * to print trace information to standard output, but a custom tracing system can
     * be implemented here as well.
     *
     */
    private void debugOut(Object o)
    {
        System.out.println("Debug: " + MAPPING_MODULE_NAME);
        if (o != null) {
            if (o.getClass().isArray()) {
                int length = Array.getLength(o);
                for (int i = 0; i < length; i++) {
                    System.out.println("\t" + Array.get(o, i));
                }
            } else {
                System.out.println("\t" + o);
            }
        }
    }
    private Subject subject;
    private CallbackHandler callbackHandler;
    private Map sharedState;
    private Map options;

    protected boolean debug = false;
}
```

Results

Using the custom login module, Microsoft Active Directory identities are mapped to the WebSphere Application Server's security registry and the behavior of the SPNEGO TAI is customized.

Single sign-on capability with SPNEGO TAI - checklist (deprecated):

WebSphere Application Server provides a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources in WebSphere Application Server. To deploy and use the SPNEGO TAI you need to examine your installation and decide on how best to configure the SPNEGO TAI.

Note:

In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

Lightweight Third Party Authentication (LTPA) is the default authentication mechanism for WebSphere Application Server. However, you may need to configure LTPA prior to configuring the SPNEGO TAI. LTPA is the required authentication mechanism for all trust association interceptors. You can configure LTPA by clicking **Security > Global security > Authentication mechanisms and expiration**.

Note: Enabling web security single sign-on (SSO) is optional when you configure the SPNEGO TAI. For more information, see “Implementing single sign-on to minimize web user authentications” on page 1441.

Answer the following questions to establish how the SPNEGO TAI is deployed.

1. What is your criteria for intercepting HTTP requests?

   You must decide if the SPNEGO TAI deployment will use the HTTPHeaderFilter class as the default. If you do use this class, then you must specify the exact filter properties for this class. The default behavior of the SPNEGO TAI is to use the com.ibm.ws.spnego.HTTPHeaderFilter class to intercept all requests.

   If you do not use the sample com.ibm.ws.spnego.HTTPHeaderFilter class, then you must define a new class that implements the com.ibm.wsspi.security.spnego.SpnegoTAIFilter interface.

   You can decide to further control what HTTP requests are intercepted using the Service Provider Programming Interface (SPI), “Filtering HTTP requests for SPNEGO TAI (deprecated)” on page 1486

   See “SPNEGO TAI custom properties configuration (deprecated)” on page 1469 for descriptions of

   - com.ibm.ws.security.spnego.SPN<id>.filterClass
   - com.ibm.ws.security.spnego.SPN<id>.filter

2. Is user Id mapping to be used? If not, why not?

   WebSphere Application Server enables you to define or develop a custom login module to map user IDs. See “Mapping Kerberos client principal name to WebSphere user registry ID for SPNEGO TAI (deprecated)” on page 1480 for more detail about performing this mapping.

   You must decide, before deploying the TAI, whether or not to use this custom login module to perform the SPNEGO TAI identity mapping

3. What type of encryption is to be used to process the SPNEGO tokens?

   Microsoft Windows Active Directory supports two different Kerberos encryption types: RC4-HMAC and DES-CBC-MD5. The IBM Java Generic Security Service (JGSS) library (and SPNEGO library) support both of these encryption types.

   Restriction: RC4-HMAC encryption is only supported with a Windows 2003 Server key distribution center (KDC).

4. How will you handle credential delegation?

   Kerberos supports the delegation of credentials. A server that receives Kerberos credentials from a client can impersonate that client to other servers by using delegated credentials. Since SPNEGO TAI

tokens are a wrapping of a Kerberos credential, a server that receives Kerberos credentials within an SPNEGO token can use those Kerberos credentials to impersonate the original user. That server can interact using SPNEGO over HTTP as a SPNEGO client to other SPNEGO servers by composing an appropriate HTTP Authorization header.

5. Will the SPNEGO TAI be deployed in a single or multiple domain name service (DNS) domain environment?

   Web browsers running on Windows are sensitive to DNS domains. They only send a SPNEGO token when the target host name identifies a host name defined in the DNS domain of the client machine. You can use HTTP redirection to support this configuration with the creation of a pseudo Kerberos service principal name (SPN) in each DNS domain. All SPNs that WebSphere Application Server supports must have their secret keys available in Kerberos `keytab` files. To enable single sign-on across multiple DNS domains, a separate Kerberos keytab file is generated for each SPN per domain. These individual Kerberos keytab files must be merged before they can be used by WebSphere Application Server.

6. How frequently will application servers reload the SPNEGO TAI properties?

   The SPNEGO TAI has an optional property reload feature that allows the reloading of the TAI properties without restarting the Java virtual machine (JVM). This reload feature is controlled by the system properties `com.ibm.ws.security.spnego.propertyReloadFile` and `com.ibm.ws.security.spnego.propertyReloadTimeout`. These properties taken together enable the SPNEGO TAI internal properties to be reloaded from a file on the file system after a certain time period. If the com.ibm.ws.security.spnego.propertyReloadTimeout attribute is set to a valid integer value, and the com.ibm.ws.security.spnego.propertyReloadFile attribute points to a file on the file system, then each JVM reloads the SPNEGO TAI properties from the file after the timeout period expires. Also, the SPNEGO TAI properties are reloaded only if the date on the file has changed. If these reload properties are not set, then the SPNEGO TAI properties are only loaded once, at JVM initialization, from the SPNEGO TAI custom properties that are defined in WebSphere Application Server configuration data. See “SPNEGO TAI JVM configuration custom properties (deprecated)” on page 1477 for more information about these reload properties.

The Windows Active Directory (Web) administrator, the WebSphere Application Server administrator, and the application team review and answer these questions to determine the best deployment and configuration settings for the SPNEGO TAI.

Filtering HTTP requests for SPNEGO TAI (deprecated):

You can use a system programming interface to customize the behavior of the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) trust association interceptor (TAI) by specifying whether or not a particular HTTP request should be intercepted.

Before you begin

Before you begin, you need to understand the deployment of the SPNEGO TAI in your installation.

Note:

> In WebSphere Application Server Version 6.1, a trust association interceptor (TAI) that uses the Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) to securely negotiate and authenticate HTTP requests for secured resources was introduced. In WebSphere Application Server 7.0, this function is now deprecated. SPNEGO web authentication has taken its place to provide dynamic reload of the SPNEGO filters and to enable fallback to the application login method.

About this task

Verify the configuration of your SPNEGO TAI. The deployment of the SPNEGO TAI can vary from a single WebSphere Application Server system on which a single application is running to a large multinode WebSphere Application Server, Network Deployment (ND) cell, with dozens of application servers, hosting many applications. Every SPNEGO TAI is installed at the cell level. You must be aware of your particular SPNEGO TAI configuration.

The default behavior of the SPNEGO TAI is to not intercept HTTP requests. This default behavior ensures that the SPNEGO TAI can be installed into an existing cell, configured for a single application server and not change any other application servers in the cell. Other WebSphere Application Servers can run exactly as before within a given configuration.

Then decide whether or not to use the sample SPN<id>.filter class and determine the exact filter properties to use.

Note: The default behavior of the SPNEGO TAI is to use the com.ibm.ws.security.spnego.SPN<id>.filter class and intercept all requests.

If the default behavior is not appropriate, you can use a customer provided class, or extend or modify the sample class as required. The system programmer interface, com.ibm.ws.security.spnego.SpnegoFilter allows you to implement a custom filter to determine whether or not to intercept a particular HTTP request. With the default implementation, you can set filter rules for coarse as well as fine-grained criteria in selecting which HTTP requests to intercept.

Procedure

1. Set the `com.ibm.ws.security.spnego.isEnabled` Java virtual machine (JVM) custom property to `true` to enable the SPNEGO TAI on any JVM.
2. Identify when the SPNEGO TAI intercepts a given request. A set of filter properties is provided, but you must determine what is appropriate and modify the `com.ibm.ws.security.spnego.SPN<id>.filter class` accordingly.

Results

Your SPNEGO TAI is set to filter HTTP requests when it is operating.

Configuring single sign-on capability with Enterprise Identity Mapping

The Enterprise Identity Mapping (EIM) identity token connection factory is a type of Java 2 Connector (J2C) connection factory. Using EIM identity token connection factories along with EIM identity token-enabled products, such as IBM Toolbox for Java, provides a single sign-on capability for WebSphere Application Server applications that need to access server data and resources through your user ID.

Before you begin

The EIM identity token connection factory is supported on the following WebSphere Application Server products.

Attention: Either Lightweight Third Party Authentication (LTPA) or Simple WebSphere Authentication Mechanism (SWAM) may be used with the EIM identity token connection factory. Enabling web security single sign-on (SSO) is optional when LTPA is used with the EIM identity token connection factory. See the information about implementing single sign-on to minimize web user authentications.

Table 97. Supported editions per product.

This table lists the supported edition names per product.

Edition name	Supported products
Version 8.0	WebSphere Application Server, Express for IBM iWebSphere Application Server (base)
Version 6.1	WebSphere Application Server, Express for IBM iWebSphere Application Server (base)
Version 6.0.x	WebSphere Application Server, Express for OS/400WebSphere Application Server (base)

This topic describes how to configure EIM identity token connection factories for Version 8.0 only and provides information about a sample application that might be helpful to you when you develop your own applications.

Attention: Configuration tasks can vary slightly for other WebSphere Application Server products and editions.

About this task

The sample application uses an EIM identity token connection factory to provide EIM identity tokens for use with IBM Toolbox for Java com.ibm.as400.access.AS400 objects. For example, if the sample application is deployed on SERVER A, you can log in once to WebSphere Application Server and use the sample application to perform IBM i server commands under your IBM i user profiles on SERVER B, SERVER C, or SERVER D.

When you make a request to the sample application, you must log in with your WebSphere Application Server user ID and password. Each request contains the server command and the target server name where the command runs. When the request is received, the application calls the connection factory to generate an identity token. The connection factory extracts your user ID from a Java Authentication and Authorization Service (JAAS) subject object provided by WebSphere Application Server security, and it collaborates with the EIM domain controller to create the identity token that is returned to the application. The application then creates a com.ibm.as400.access.AS400 object for SERVER B and provides it with the identity token (instead of your IBM i user profile) before it passes the server command to run.

Attention: A new identity token and com.ibm.as400.access.AS400 object are created each time you send a request that contains a new target server. All com.ibm.as400.access.AS400 objects are stored in an HTTP Session for use with subsequent requests.

Procedure

1. Verify that you have all of the necessary prerequisites installed to use the EIM token connection factory. You must verify that you have installed the necessary program temporary fixes (PTF) to your server and applications. For more information, see “Verifying Enterprise Identity Mapping identity token connection factory prerequisite applications” on page 1489.
2. Configure EIM work with the identity token connection factory. These instructions explain how to complete the following tasks:
   a. Create a domain in EIM.
   b. Add the domain to domain management.
   c. Create a source user registry definition.
   d. Create a user identifier.
   e. Create a target association.
   f. Create a source association.
   g. Test the connection to the EIM domain controller

   For more information, see “Configuring Enterprise Identity Mapping” on page 1489.

3. Configure the EIM identity token connection factory. This step involves configuring two Java Archive (JAR) files and a shared library. For more information, see “Configuring the Enterprise Identity Mapping identity token connection factory” on page 1492.
4. Configure the connection factory. For more information, see “Automatically configuring the connection factory” on page 1497.

Results

After completing the previous steps, you have configured single sign-on for Enterprise Identity Mapping.

Verifying Enterprise Identity Mapping identity token connection factory prerequisite applications:

Use the following procedure to verify that the necessary prerequisites have been installed before using the Enterprise Identity Mapping (EIM) identity token connection factory.

Before you begin

Before you can use the EIM identity token connection factory, you must have the required cumulative program temporary fix (PTF) applied to your server. You might also need to apply PTFs to run the sample application.

About this task

Perform the following steps to verify that you have the necessary prerequisites installed to use the EIM identity token connection factory:

Procedure

1. Verify that OS/400 - Extended Base Directory Support (5761–SS1 or 5770-SS1 option 3) is installed on the IBM i system that hosts WebSphere Application Server. This product is a requirement for the EIM Identity Token Connection Factory.
2. Verify that the latest operating system PTFs are applied to your IBM i system for the EIM Identity Token Connection Factory. For information about the latest operating system PTFs, search the Technotes for "EIM Identity Token Connection Factory" on the WebSphere Application Server support page at: http://www.ibm.com/software/webservers/appserv/support.html.
3. Install the required PTFs for the sample application.

   To run the sample application, verify that the latest IBM Toolbox for Java service packs and the required operating system PTFs are installed on all of your iSeries servers. For information on obtaining the latest service packs and required operating system PTFs, see Toolbox for Java and JTOpen Service Packs.

Results

After verifying that you have the necessary prerequisite applications installed, you can configure EIM for use with the identity token connection factory.

What to do next

Configure EIM. See “Configuring Enterprise Identity Mapping.”

Configuring Enterprise Identity Mapping:

Use the iSeries Navigator to configure Enterprise Identity Mapping (EIM) for use with the identity token connection factory.

Before you begin

For these steps, assume that your EIM controller, which is your Lightweight Directory Access Protocol (LDAP) directory server, is your local directory server and that it resides on the iSeries server that is being configured for EIM. For detailed information about EIM, see “Enterprise Identity Mapping” on page 1440.

You need the LDAP server administrator distinguished name (DN) and password to perform this task.

Tip: A server can participate only in one EIM domain at a time. If your server is already joined to an EIM domain and the domain is added to domain management, use that domain, and skip to Create a source user registry definition in EIM.

Procedure

1. The identity token connection factory requires you to configure an EIM domain.
   Create a domain in EIM:

   Note: Depending on the setup of the machine, these steps might appear in a slightly different order. This assumes that LDAP is already configured and the network authentication service has not been configured.

   a. Make sure that the LDAP server started. You can verify the LDAP server administrator distinguished name (DN) and password. However, be aware that the LDAP server is stopped by the wizard later on.
   b. In iSeries Navigator, expand ***server_name*** **> Network > Enterprise Identity Mapping**, where *server_name* is the name of your iSeries server.
   c. Click **Enterprise Identity Mapping**.
   d. Right-click **Configuration** and select **Configure** to start the EIM Configuration wizard.

      Note: This option is labeled **Reconfigure** if EIM has been previously configured on the system.

   e. On the Welcome page of the wizard, select **Create and join a new domain**.
   f. Click **Next**.
   g. On the Specify EIM Domain Location page, select **On the local Directory server** and then click **Next**.
   h. If the network authentication service has not been configured on the system to set up a single sign-on environment, the Configure Network Authentication Service page is displayed. Network Authentication Service is not required for the EIM identity token connection factory. Select **No** and then click **Next**.
   i. On the Specify User for Connection page, specify the distinguished name and password for the LDAP administrator to ensure that the wizard has enough authority to administer the EIM domain and the objects in it. Click **Next**.

      Note: If you have not configured the local directory server before you use the EIM Configuration wizard, the Configure Directory Server page displays instead. Use this page to specify the distinguished name and password for the LDAP administrator and continue with the next step in this procedure. The LDAP distinguished name (DN) identifies the LDAP administrator for the directory server. The EIM Configuration wizard creates this LDAP administrator DN and uses it to configure the directory server as the domain controller for the new domain that you are creating.

   j. On the Specify Domain page, provide the name of the EIM domain, and click **Next**.
   k. On the Specify Parent DN for Domain page, select **Yes** to specify a parent DN for the domain that you are creating, or specify **No** to have EIM data stored in a directory location with a suffix whose name is derived from the EIM domain name. Click **Next**.
   l. A message is displayed that indicates that you must stop the LDAP server. Click **Yes** to continue.

m. On the Registry Information page, select **Local OS/400** and then click **Next**.

n. On the Specify EIM System User page, select **Distinguished name and password** as the user type, provide the DN and password for the directory server administrator, and optionally, verify the DN and password. Click **Next**.

o. In the Summary panel, review the configuration information that you have provided. If all information is correct, click **Finish**.

2. Add the domain to domain management:

   a. In the iSeries Navigator, expand ***system_name*> Network > Enterprise Identity Mapping > Domain Management**.

   b. Right-click **Domain Management** and then select **Add Domain**.

   c. In the Add Domain dialog, specify the domain you created earlier and click **OK**.

3. Create a source user registry definition in EIM.

   The identity token connection factory requires a source user registry definition entry in EIM. The source user registry definition represents the registry that WebSphere Application Server uses for authentication. This registry can be a local OS registry or an LDAP registry.

   a. In iSeries Navigator, expand ***system_name* > Network > Enterprise Identity Mapping > Domain Management > *domain_name*> User Registries**.

   b. If you are prompted for the LDAP server password, provide the password and click **OK**.

   c. Right-click **User Registries** and select **Add Registry** > **System** to start the configuration wizard that adds the registry to your domain.

      Provide the registry name and type. If your application server is hosted on an iSeries server and configured to use the local OS user registry, select **OS/400** as the EIM user registry type. If your application server is configured to use the LDAP user registry, enter `LDAP - short name` as the EIM registry type.

      Note: Prior to IBM i V5R4, instead of `LDAP - short name` use `1.3.18.02.33.14-caseIgnore`. The value `1.3.18.02.33.14-caseIgnore` is the ObjectIdentifier-normalization form of the user registry type and principals are identified by the LDAP short name attribute. The wizard does not handle the descriptive name for this registry type.

   d. Click **OK**.

4. Create user identifier in EIM

   The identity token connection factory requires a user identifier entry, which is equivalent to an EIM identifier; in EIM, the user identifier entry represents the user of the application.

   a. In iSeries Navigator, expand ***system* > Network > Enterprise Identity Mapping > Domain Management > *domain* > Identifiers.**

   b. Right-click **Identifiers**, and select **New Identifier**.

   c. Enter an identifier name, such as your full name, and click **OK**.

5. Create a target association in EIM for the user identifier.

   A target association represents the user profile on the target iSeries server for the identifier created earlier.

   a. In iSeries Navigator, expand ***system* > Network > Enterprise Identity Mapping > Domain Management > *domain* > Identifiers**.

   b. Double-click the **Application Identifier** for the user created previously.

   c. Click the **Associations** tab.

   d. Click **Add**.

   e. Provide the IBM i user profile for the EIM identifier in the User field and click **OK**.

   f. Click **OK** to save the association.

6. Create a source association in EIM for the user identifier.

   A source association is used to authenticate to WebSphere Application Server.

a. In iSeries Navigator, expand ***system*** > **Network** > **Enterprise Identity Mapping** > **Domain Management** > ***domain*** > **Identifiers**.
b. Double-click the **Application Identifier** for the user created previously.
c. Click the **Associations** tab.
d. Click **Add**.
e. Click **Browse** and select the WebSphere Application Server user registry.
f. Specify your WebSphere Application Server user ID, such as `my_id`.
g. Select **Source**.
h. Click **OK** to add the new association.
i. Click **OK** to save the association.

7. Optional: Test the connection to the EIM domain controller.

   Use the **idsldapsearch** command to test the connection to the EIM domain controller. For example, if the LDAP server is located on the `my_server` host, the EIM domain name is `My_EIM_Domain`, and the source user registry is `WAS Registry`, the steps to test the connection are as follows:

   a. Log on to the iSeries server that hosts your WebSphere Application Server profile.
   b. From a CL command line, specify `QSH` and press Enter.
   c. Specify the following command and press Enter:

```
idsldapsearch -h my_server -p 389 -D cn=administrator
-w secret -b "ibm-eimDomainName=My_EIM_Domain"
"ibm-eimRegistryName=WAS_Registry"
```

   where:

   - *my_server* is the name of the host server of the LDAP server.
   - `389` is the port that is used by the LDAP server.
   - `cn=administrator` is the LDAP DN of the LDAP administrator.
   - `secret` is the LDAP administrator password.
   - `ibm-eimDomainName=My_EIM_Domain` is the LDAP DN of the EIM domain name entry.

   The previous lines display as multiple lines for illustrative purposes only. Specify the command as one continuous line.

   In this example, no EIM domain parent name exists. If an EIM domain parent name did exist, such as dc=myserver,dc=ibm,dc=com, the LDAP DN is `ibm-eimDomainName=My_EIM_Domain,dc=myserver,dc=ibm,dc=com`.

Results

The expected output looks similar to the following example:

```
ibm-eimRegistryName=WAS Registry,cn=Registries,ibm-eimdomainname=My_EIM_Domain
   objectclass=top
   objectclass=ibm-eimRegistry
   objectclass=ibm-eimSystemRegistry
   ibm-eimRegistryName=WAS_Registry
   ibm-eimRegistryType=1.3.18.0.2.33.9-caseIgnore
   description=Example Registry for WebSphere Application Server
```

What to do next

Configure the EIM identity token connection factory. See “Configuring the Enterprise Identity Mapping identity token connection factory.”

Configuring the Enterprise Identity Mapping identity token connection factory:

The Enterprise Identity Mapping (EIM) identity token connection factory requires the `eim.jar` file to be located in the class path for the connection factory. The `jt400.jar` file must be in the class path for the sample application.

About this task

Perform the following steps to configure the `eim.jar` and `jt400.jar` files:

Procedure

1. “Configuring the eim.jar and jt400.jar files” on page 1494
2. “Configuring a shared library for the jt400.jar file” on page 1495

Enterprise Identity Mapping identity token connection factory parameters:

The following table is a summary of the parameters or custom properties that are referenced by the Enterprise Identity Mapping (EIM) identity token connection factory. These parameters are necessary when you configure the EIM identity token connection factory.

Table 98. Parameters and custom properties referenced by EIMidentity token connection factory.

This table lists the parameters and custom properties referenced by EIMidentity token connection factory.

Parameter description	Parameter example	Required	Initially set by	Referenced by
LDAP administrator ID and password	`cn=administrator`	Yes	LDAP administrator using the iSeries Navigator when configuring LDAP	J2C Authentication Data entry
LDAP host name and port	`mysystem.com and 389`	Yes	LDAP administrator using the iSeries Navigator	LdapHostName and LdapHostPort identity token resource adaptor properties
EIM domain name and parent domain	`EIM` and `dc=mysystem,dc=com`	Yes	EIM administrator using the iSeries Navigator when configuring EIM	EimDomainName and ParentDomain identity token resource adaptor properties
sourceRegistryName	`LDAP`	Yes	EIM administrator using the iSeries Navigator when configuring EIM user registries that are used by applications	sourceRegistryName identity token resource adaptor property
Key time out and size	`1200` and `512`	No	WebSphere Application Server administrator using the administrative console	KeyTimeoutSeconds and KeySize identity token resource adaptor properties
UseSSL	`false`	No	WebSphere Application Server administrator using the administrative console	UseSSL identity token resource adaptor property
TrustStoreName	`profile_root/etc/idtokTrustFile.jks`	No	WebSphere Application Server administrator using the administrative console	TrustStoreName identity token resource adaptor property

Table 98. Parameters and custom properties referenced by EIMidentity token connection factory (continued).

This table lists the parameters and custom properties referenced by EIMidentity token connection factory.

Parameter description	Parameter example	Required	Initially set by	Referenced by
TrustStorePassword	`tspwd`	No	WebSphere Application Server administrator using the administrative console	TrustStorePassword identity token resource adaptor property
KeyStoreName	`profile_root/etc/idtokKeyFile.jks`	No	WebSphere Application Server administrator using the administrative console	KeyStoreName identity token resource adaptor property
KeyStorePassword	`kspwd`	No	WebSphere Application Server administrator using the administrative console	KeyStorePassword identity token resource adaptor property

Identity token files

After applying the required PTFs, all of the files in the table below can be found on the server where you have WebSphere Application Server installed.

Table 99. Files found after required PTFs are applied.. This table lists the files found after required PTFs are applied.

File Name	Directory
`idTokenRA.rar`	`/QIBM/ProdData/OS400/security/eim`
`testIdentityToken.ear`	`/QIBM/ProdData/OS400/security/eim`
`cfgIdToken.jacl`	`/QIBM/ProdData/OS400/security/eim`
`eim.jar`	`/QIBM/ProdData/OS400/security/eim`
`jt400.jar`	`/QIBM/ProdData/HTTP/public/jt400/lib`
`idTokenRA.JCA15.rar`	`/QIBM/ProdData/OS400/security/eim`

Configuring the eim.jar and jt400.jar files:

You can configure the EIM identity token factory by using the following procedure.

About this task

Completing the steps in this topic is the first part of configuring the EIM identity token connection factory.

Procedure

The `eim.jar` file is already configured on your iSeries server and no additional action is required.

Results

The `eim.jar` and the `jt400.jar` files are configured.

What to do next

If you copy the jt400.jar file to a different directory, you must configure a shared library for the file. For OS/400 or IBM i, JTOpen Version 4.3 or later of the jt400.jar file is already on your server. However, you still must configure a shared library for the jt400.jar file. See the “Configuring a shared library for the jt400.jar file” topic for more information.

Configuring a shared library for the jt400.jar file:

Use the WebSphere Application Server administrative console to create a shared library for the jt400.jar file.

About this task

If you copied the jt400.jar file to a different directory, completing the steps in this topic is part of configuring the Enterprise Identity Mapping (EIM) token connection factory.

Procedure

1. Create a shared library:
   a. In the WebSphere Application Server administrative console, expand **Environment**.
   b. Click **Shared Libraries**.
   c. Click to expand the **Scope** field.
   d. Select the node where you want to create the shared library.
   e. Click **Apply**.
   f. Click **New**.
   g. Specify the name of the shared library in the **Name** field.
   h. Specify the full path name of the jt400.jar file in the **Classpath** field. The default path name is /QIBM/ProdData/HTTP/public/jt400/lib/jt400.jar.
   i. Click **OK**.
2. Create an application class loader for the shared library. This step makes the jt400.jar file available to all applications that are deployed on the application server.
   a. In the WebSphere Application Server administrative console, click **Servers > Application servers > *server_name***.
   b. Under the Server Infrastructure heading, click **Java and Process Management > Class loader > New**.
   c. Keep the Class loader order default as **Classes loaded with parent class loader first** and click **OK**.
   d. Click the Class loader ID for the class loader that was created.
   e. Under Additional properties, click **Shared library references**.
   f. Click **Add**.
   g. Select the name of the shared library you created earlier.
   h. Click **OK**.
3. Grant the java.security.AllPermission permission to the jt400.jar file in the server.policy file.

   To grant the required permission to the jt400.jar file, edit the server.policy file for your WebSphere Application Server profile and add the following statement. The server.policy file is in the *profile_root*/properties directory.

   ```
   grant codeBase "file:path_name/jt400.jar" {
     permission java.security.AllPermission;
   };
   ```

   where *path_name* is the fully qualified path name of the directory that contains the jt400.jar file. The default path name is /QIBM/ProdData/HTTP/public/jt400/lib/jt400.jar.

4. Save your configuration changes.
   a. Expand **System administration** and click **Save Changes to Master Repository**.
   b. Click **Save**.

Results

The shared library for the `jt400.jar` file is configured.

What to do next

After completing these steps, continue with configuring the connection factory. See “Manually configuring the connection factory” to configure the connection factory manually, or see “Automatically configuring the connection factory” on page 1497 to use a Jacl script to automatically configure the connection factory.

Manually configuring the connection factory:

The following steps help you manually configure the connection factory.

Before you begin

Configure the `eim.jar` and `jt400.jar` files.

About this task

After you configure the `eim.jar` and `jt400.jar` files, you can choose to manually or automatically configure the connection factory. If you choose to automatically configure the connection factory, see “Automatically configuring the connection factory” on page 1497 for more information. Perform the following steps to manually configure the Java 2 Connector (J2C) authentication data, the resource adapter, and the connection factory.

Procedure

1. Configure the Java 2 Connector (J2C) authentication data.
   a. In the WebSphere Application Server administrative console, click **Security >Global security**.
   b. Under Java Authentication and Authorization Service, click **J2C Authentication data > New**
   c. Specify the values for each of the required fields. The User ID (cn=administrator for example) and Password values are those that are used by the connection factory to bind to the Lightweight Directory Access Protocol (LDAP) server that contains your Enterprise Identity Mapping (EIM) data.
   d. Click **OK**.
2. Configure the resource adapter.
   a. In the WebSphere Application Server administrative console, click **Resources > Resource adapters > Resource adapters**.
   b. Select the node where you want to install the resource adapter.
   c. Click **Apply**.
   d. Click **Install RAR**.
   e. Select **Local path** if you have a drive that is mapped to your iSeries server. Otherwise, select **Server path**.
   f. Specify the path name or browse to the path name for the `idTokenRA.JCA15.rar` RAR file.
   g. Click **Next**.
   h. Specify the name of your adapter in the Name field. For example, specify `identitytoken`.
   i. Click **OK**.
3. Configure the connection factory.

a. On the Resource Adapters panel, click the name of your newly created resource adapter.
b. Under Additional Properties, click **J2C connection factories > New**.
c. Specify the name of your connection factory in the Name field. For example, specify `idtokenconnection`.
d. Specify `eis/IdentityToken` in the Java Naming and Directory Interface (JNDI) name field. This name must match the JNDI name used during the deployment of the sample application. The name is used for reference binding.
e. In the Component-managed authentication alias and Container-managed authentication alias fields, select the authentication data alias that you created earlier.
f. In the Mapping-configuration alias field, select **DefaultPrincipalMapping**.
g. Click **Apply**.
h. Under Additional Properties, click **Custom properties**. The custom properties are used by the connection factory to communicate with the EIM controller. View the custom property descriptions, and determine whether the properties are required or optional. For more information, see "Enterprise Identity Mapping identity token connection factory parameters" on page 1493.

   To set a property value, complete the following steps:

   1) Click the name of the custom property.
   2) Type the value of the property in the Value field.
   3) Click **OK**.

4. Save your configuration changes.
   a. Expand **System administration** and click **Save Changes to Master Repository**.
   b. Click **Save**.

Results

You have manually configured the connection factory.

What to do next

After saving your configuration changes, you can deploy the EIM sample application into the WebSphere Application Server environment. The source code files that are used in the sample application can be used as a model for creating your own applications. See "Deploying the Enterprise Identity Mapping sample application" on page 1498 for more information.

Automatically configuring the connection factory:

You can use the `cfgIdToken.jacl` script to automatically configure the Java 2 Connector (J2C) authentication data, the resource adapter, and the connection factory.

Before you begin

Configure the `eim.jar` and `jt400.jar` files.

About this task

After you configure the `eim.jar` and the `jt400.jar` files, you can choose to manually or automatically configure the connection factory. If you choose to manually configure the connection factory, see "Manually configuring the connection factory" on page 1496 for more information.

Perform the following steps to create a connection factory named CF1 in the `my_profile` WebSphere Application Server profile:

Procedure

1. Verify that your application server is started.
2. On the CL command line, enter `QSH`. This command starts the Qshell environment.
3. Change to the *app_server_root*`/bin` directory and specify the following command:

```
wsadmin -profileName my_profile -f /QIBM/ProdData/OS400/security/eim/cfgIdToken.jacl
  CF1 sys1.ibm.com 389 "Eim Domain 1"  "Registry For my_profile"
  -rarFile /QIBM/ProdData/OS400/security/eim/idTokenRA.JCA15.rar  -authAlias myAlias1
  -authUserName cn=administrator -authPassword pwd1
```

   Note: The `/QIBM/ProdData/OS400/security/eim` directory contains two resource adapter archive files, `idTokenRA.rar` and `idTokenRA.JCA15.rar`. The resource adapter contained in `idTokenRA.rar` is implemented to the Java EE Connector Architecture (JCA) 1.0 specification, while the adapter in `idTokenRA.JCA15.rar` is implemented to the JCA 1.5 specification. The JCA 1.5 specification is included in the Java EE 1.4 specification.

   where:

   - *my_profile* is the name of the WebSphere Application Server profile.
   - `/QIBM/ProdData/OS400/security/eim/cfgIdToken.jacl` is the path name to the `cfgIdToken.jacl` script.
   - `CF1` is the name of the connection factory.
   - *sys1.ibm.com* is the Lightweight Directory Access Protocol (LDAP) server host name for the Enterprise Identity Mapping (EIM) domain controller.
   - `389` is the LDAP server port.
   - *Eim Domain 1* is the EIM domain name.
   - *Registry For my_profile* is the EIM source user registry.
   - `/QIBM/ProdData/OS400/security/eim/idTokenRA.JCA15.rar` is the path name to the `idTokenRA.JCA15.rar` file.
   - *myAlias1* is the authentication alias name that is referenced by the connection factory when it authenticates to the EIM domain controller (LDAP server).
   - `cn=administrator` is the distinguished name that is associated with the authentication alias.
   - `pwd1` is the password that is associated with the authentication alias.

   Notes®:

   - The previous sample displays on multiple lines for illustrative purposes only. Type the command on one continuous line.
   - Quote all argument values that contain embedded blanks.

Results

You have automatically configured the connection factory.

What to do next

After performing the previous steps, you can deploy the EIM sample application into the WebSphere Application Server environment. The source code files that are used in the sample application can be used as a model for creating your own applications. See “Deploying the Enterprise Identity Mapping sample application” for more information.

Deploying the Enterprise Identity Mapping sample application:

You can deploy the sample application into the WebSphere Application Server environment.

Before you begin

Using Enterprise Identity Mapping (EIM) identity token connection factories requires that WebSphere Application Server administrative security be enabled. However, no restrictions or limitations exist on how you choose to configure administrative security.

Before you deploy the sample application, you must enable WebSphere Application Server administrative security. This step is not required if you already have administrative security enabled for your WebSphere Application Server profile. For more information on how to configure security, see “Enabling security” on page 1172.

About this task

The source code files that are used to implement the sample application are contained in the `testIdentityToken.ear` file and can be used as a model for creating your own applications.

The com.ibm.identitytoken.IdentityTokenTest class is a servlet in the sample application. After the application is deployed, the source code file for the IdentityTokenTest servlet is in this directory:

profile_root/installedApps/testIdentityToken.ear/testIdentityTokenWeb.war
/WEB-INF/source/com/ibm/identityToken/IdentityTokenTest.java

Note the IdentityTokenTest servlet design features when you implement your own application.

- A profile variable with a String type and the name, sourceApplicationID, is set in the init method of the IdentityTokenTest servlet. This variable is later used with the setSourceApplicationID method of a ConnectionSpecImpl object to uniquely identify the application to Enterprise Identity Mapping (EIM). When you implement your own applications, use a similar convention to assign a unique SourceApplicationID ID.
- After an identity token is generated, it is used to create a com.ibm.as400.access.AS400 object, which is stored in an HTTPSession object immediately after the AS400 object is used to run the OS/400 server command on the selected host server. Only the AS400 object persists across requests to the server (not the IdentityToken object), which provides improved performance for subsequent requests, and the identity token does not expire.

The following steps help you deploy the sample application into the WebSphere Application Server environment.

Procedure

1. Restart your application server.
2. Deploy the sample application.
   a. In the WebSphere Application Server administrative console, click **Applications > Install applications**.
   b. Select **Local path** if you have a drive mapped to your iSeries server. Otherwise, select **Server path**.
   c. Specify the path name or browse to the path name for the `testidentitytoken.ear` enterprise archive (EAR) file. This file is found in the `/QIBM/ProdData/OS400/security/eim/` directory on your server.
   d. Click **Next**.
   e. Optional: Change the virtual host values.
   f. Click **Next**.
   g. Select your installation options, and click **Next**.
   h. Decide whether to map modules to servers and click **Next**.
   i. Select your module in the Map resource references to resources panel and click **Next**.

j. Optional: Change the Java Naming and Directory Interface (JNDI) name for the `eis/IdentityToken_Shared_Reference` reference binding . Do this step if you configured your connection factory with a JNDI name other than `eis/IdentityToken`.

k. Accept the default values for the remainder of the panels and click **Next**.

l. On the Summary panel, click **Finish**.

m. Expand **System administration** and click **Save Changes to Master Repository**.

n. Click **Save**.

3. Run the sample application.

a. In the WebSphere Application Server administrative console, click **Applications > Enterprise applications**.

b. Select the **testIdentityToken** application.

c. Click **Start**.

d. Open a new session of your web browser.

e. If you mapped the sample application web module to an external web server, refresh your WebSphere Application Server web server plug-in.

To refresh the web server plug-in, perform the following steps:

1) Click **Servers > Web servers > *Web_server_name***.

2) Click **Generate Plug-in**.

f. Specify the application welcome page from your web browser. Use the following web address:

```
http://your.server.name:port/testIdentityTokenWeb/IDTknTest.jsp
```

The *your.server.name* and *port* variables are the values for your external web server or internal HTTP transport (WebSphere Application Server container).

g. Specify a value for OS/400 host system name and for OS/400 command. For example, if you have EIM configured for the my_server server, specify `my_server` in the **OS/400 host system name** field. Specify `crtlib my_library` in the **OS/400 command** field.

h. Click **Submit**.

i. Specify a user ID and password at the login prompt.

After you click **Submit**, the request is sent to the IdentityTokenTest servlet, which is protected by the allUsers role. The allUsers role is bound to the AllAuthenticated special subject so any user in the WebSphere Application Server user registry is authorized to access the IdentityTokenTest servlet.

j. Click **OK**. If you specified `my_library`, the response is similar to the following example:

```
Library my_library created.
```

k. Verify that the library is created under the user profile that is mapped by EIM:

1) From a CL command line, enter `wrklnk '/QSYS.LIB/my_library.lib'`.

2) On the Work with Object Links screen, enter `8` in the option field to the left of `my_library.lib`.

3) Verify that the value of the Owner attribute for the `my_library` library is the user profile that is mapped by EIM.

Configuring single sign-on capability with Tivoli Access Manager or WebSEAL

Use the following information to enable single sign-on to WebSphere Application Server using either WebSEAL or the plug-in for web servers.

About this task

Either Tivoli Access Manager WebSEAL or Tivoli Access Manager plug-in for web servers can be used as reverse proxy servers to provide access management and single sign-on (SSO) capability to WebSphere Application Server resources. With such an architecture, either WebSEAL or the plug-in authenticates users and forwards the collected credentials to WebSphere Application Server in the form of an IV Header. Two types of single sign-on are available, the TAI interface and the TAI++ interface, so named as both use

WebSphere Application Server trust association interceptors (TAI). With the TAI, the end-user name is extracted from the HTTP header and forwarded to embedded Tivoli Access Manager where the end-user name is used to construct the client credential information and authorize the user. With the TAI++, all of the user credential information is available in the HTTP header and not just the user name. The TAI++ is the more efficient of the two solutions because a Lightweight Directory Access Protocol (LDAP) call is not required. TAI functionality is retained for backwards compatibility.

Complete the following tasks to enable single sign-on to WebSphere Application Server using either WebSEAL or the plug-in for web servers. These tasks assume that embedded Tivoli Access Manager is configured for use.

Procedure

1. Create a trusted user account for Tivoli Access Manager in the shared Lightweight Directory Access Protocol (LDAP) user registry. For more information, see “Creating a trusted user account in Tivoli Access Manager” on page 1507.
2. Configure either WebSEAL or the Tivoli Access Manager plug-in for Web servers to work with WebSphere Application Server. For more information, see either of the following articles:
   - “Configuring WebSEAL for use with WebSphere Application Server” on page 1507
   - “Configuring Tivoli Access Manager plug-in for web servers for use with WebSphere Application Server” on page 1508
3. Configure single sign-on using either the TAI or TAI++ interface. For more information, see either of the following articles:
   - “Configuring single sign-on using trust association” on page 1509
   - “Configuring single sign-on using trust association interceptor ++” on page 1510

Single sign-on settings:

Use this page to set the configuration values for single sign-on (SSO).

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under Authentication, click **Web and SIP security > Single sign-on (SSO)**.

Note: The Set security cookies as HTTPOnly to resist cross-site scripting attacks check box has been added to the Single sign-on settings page for this release. The HttpOnly attribute is a browser attribute created to prevent client side applications (such as Java scripts) from accessing cookies to prevent some cross-site scripting vulnerabilities. The attribute specifies that LTPA and WASReqURL cookies include the HTTPOnly field.

Enabled:

Specifies that the single sign-on function is enabled.

Web applications that use J2EE FormLogin style login pages, such as the administrative console, require single sign-on (SSO) enablement. Only disable SSO for certain advanced configurations where LTPA SSO-type cookies are not required.

Data type:	Boolean
Default:	Enabled
Range:	Enabled or Disabled

Requires SSL:

Specifies that the single sign-on function is enabled only when requests are made over HTTPS Secure Sockets Layer (SSL) connections.

Data type: Boolean
Default: Disable
Range: Enable or Disable

Domain name:

Specifies the domain name (.ibm.com, for example) for all single sign-on hosts.

The application server uses all the information after the first period, from left to right, for the domain names. If this field is not defined, the web browser defaults the domain name to the host name where the web application is running. Also, single sign-on is then restricted to the application server host name and does not work with other application server host names in the domain.

You can specify multiple domains separated by a semicolon (;), a space ( ), a comma (,), or a pipe (|). Each domain is compared with the host name of the HTTP request until the first match is located. For example, if you specify ibm.com;austin.ibm.com and a match is found in the ibm.com domain first, the application server does not match the austin.ibm.com domain. However, if a match is not found in either ibm.com or austin.ibm.com, then the application server does not set a domain for the LtpaToken cookie.

If you specify the `UseDomainFromURL` value, the application server sets the SSO domain name value to the domain of the host that is used in the web address. For example, if an HTTP request comes from `server1.raleigh.ibm.com`, the application server sets the SSO domain name value to `raleigh.ibm.com`.

Tip: The UseDomainFromURL value is case insensitive. You can type `usedomainfromurl` to use this value.

Data type: String

Interoperability mode:

Specifies that an interoperable cookie is sent to the browser to support back-level servers.

In WebSphere Application Server, Version 6 and later, a new cookie format is needed by the security attribute propagation functionality. When the interoperability mode flag is enabled, the server can send a maximum of two single sign-on (SSO) cookies back to the browser. In some cases, the server just sends the interoperable SSO cookie.

Web inbound security attribute propagation:

When web inbound security attribute propagation is enabled, security attributes are propagated to front-end application servers. When this option is disabled, the single sign-on (SSO) token is used to log in and recreate the Subject from the user registry.

With this information, the receiving server can contact the originating server using an MBean call to get the original serialized security attributes.

Set security cookies as HTTPOnly to resist cross-site scripting attacks:

The HttpOnly attribute is a browser attribute created to prevent client side applications (such as Java scripts) from accessing cookies to prevent some cross-site scripting vulnerabilities. The attribute specifies that LTPA and WASReqURL cookies include the HTTPOnly field.

For session cookies, see the session settings for servers, applications, and web modules.

Data type:	boolean
Default:	enabled
Range:	enabled or disabled

com.tivoli.pd.jcfg.PDJrteCfg utility for Tivoli Access Manager single sign-on:

The com.tivoli.pd.jcfg.PDJrteCfg utility configures the Java Runtime Environment component for Tivoli Access Manager. This utility enables Java applications to use the Tivoli Access Manager policy and authorization servers.

Purpose

Steps

To run the pdjrtecfg script, perform the following steps:

1. Log into your system with a user profile and the all object (*ALLOBJ) authority.
2. On the command line, enter the Start Qshell (STRQSH) command.
3. Change to the `/bin` subdirectory of WebSphere Application Server. For example:

   `cd` *`app_server_root`*`Base/bin`
4. Run the script. For example:

```
 pdjrtecfg -action config -profileName myprofile
-host mypolicy.mycompany.com -config_type full
```

   The previous example was split onto multiple lines for illustrative purposes only.

Syntax

The following syntax diagram shows the usage of the pdjrtecfg script:

```
pdjrtecfg
    -action config
          -profileName profile_name
          -host policy_server_name
          -config_type { full | standalone }
          -cfgfiles_path configuration_file_path
    -action unconfig
          -profileName profile_name
```

Parameters

`-action {config|unconfig}`
Specifies the action to be performed. Actions include:

config Use to configure the Access Manager Java Runtime Environment component.

unconfig
Use to reconfigure the Access Manager Java Runtime Environment component.

`-cfgfiles_path`
Specifies where the generated configuration files will be placed.

Note: This parameter is required.

`-config_type {full|standalone}`
Specifies the configuration type of Java Runtime Environment for Tivoli Access Manager. Specify `full` or `standalone` with this argument. This option is required.

`-host` *`policy_server_host`*
Specifies the policy server host name.

Valid values for *policy_server_host* include any valid IP host name.

Examples include:

```
host = libra
host = libra.dallas.ibm.com
```

Notifies Tivoli Access Manager Runtime for Java that the WebSphere Application Server version is being configured so it is not necessary to perform certain steps such as copying the Java security jar files and PD.jar file since they were already placed in the appropriate directory by the WebSphere Application Server installer.

-profileName
Specifies the name of the WebSphere Application Server profile. If not specified, the default profile is used.

Specifies the fully qualified path to the Java runtime (such as the directory ending in jre). If this parameter is not specified, the home directory for the jre in the PATH statement is used. If the home directory for the jre is not in the PATH statement, this utility can create an incorrect parameter in the output files.

Comments

This command copies Tivoli Access Manager Java libraries to a library extensions directory that exists for a Java runtime that has already been installed on the system.

You can install more than one Java Runtime Environment (JRE) on a given machine. The `pdjrtecfg` command can be used to configure the Tivoli Access Manager Java Runtime Environment component independently for each of the JRE configurations.

com.tivoli.pd.jcfg.SvrSslCfg utility for Tivoli Access Manager single sign-on:

The utility is used to configure and remove the configuration information associated with WebSphere Application Server and the Tivoli Access Manager server.

Purpose

The svrsslcfg script creates a user account and server entries that represent your WebSphere Application Server profile in the Tivoli Access Manager user registry. In addition, a configuration file and a Java keystore file, which securely stores a client certificate, are created in the application server profile. This client certificate permits callers to use Tivoli Access Manager authentication services. You can also choose to remove the user and server entries from the user registry and clean up the local configuration and keystore files.

The svrsslcfg script wraps the SvrSslCfg class and provides support for multiple WebSphere Application Server profiles. The use of multiple profiles allows you to create multiple WebSphere Application Server environments that are completely isolated from one another.

Steps

To run the svrsslcfg script, perform the following steps:

1. Log on with a user profile and all object (*ALLOBJ) authority.
2. On the CL command line, enter the `Start Qshell (STRQSH)` command.
3. Change directories to the *app_server_root*`/bin` directory.
4. Enter the svrsslcfg command with the options that you want.

   For example:

```
svrsslcfg -profileName myprofile -action config -admin_id sec_master
  -admin_pwd pwd123 -appsvr_id ibm9 -appsvr_pwd ibm9pwd -mode remote
  -port 8888 -policysvr ourserv.rochester.ibm.com:7135:1
  -authzsvr ourserv.rochester.ibm.com:7136:1
  -key_file profile_root/myprofile/etc/ibm9.kdb
  -cfg_action create
```

The previous example displays on multiple lines for illustrative purposes only.

Syntax

The configuration syntax is:

```
svrsslcfg -action config
          [ -profileName profile_name ]
            -admin_id admin_user_id
            -admin_pwd admin_password
            -appsvr_id application_server_name
            -port port_number
            -mode { local | remote }
            -policysvr policy_server_name
            -authzsvr authorization_server_name
            -key_file fully_qualified_name_of_key_file
            -appsvr_pwd application_server_password
            -cfg_action { create | replace }
          [ -domain Tivoli_Access_Manager_domain ]
```

The unconfigure syntax is:

```
svrsslcfg -action unconfig
          [ -profileName profile_name ]
            -admin_id admin_user_id
            -admin_pwd admin_password
            -appsvr_id application_server_name
            -policysvr policy_server_name
          [ -domain Tivoli_Access_Manager_domain ]
```

You can enter the previous syntax as one continuous line.

Parameters

-action {config | unconfig}
Specifies the configuration action that is performed by the script. The following options apply:

-action config
Configuring a server creates user and server information in the user registry and creates local configuration and key store files on the application server. Use the `-action unconfig` option to reverse this operation.

If this action is specified, the following options are required: `-admin_id`, `-admin_pwd`, `-appsvr_id`, `-port`, `-mode`, `-policysvr`, `-authzsvr`, and `-key_file`.

-action unconfig
Reconfigures an application server to complete the following actions:

- Remove the user and server information from the user registry
- Delete the local key store file
- Remove information for this application from the configuration file without deleting the file

The reconfiguration operation fails only if the caller is unauthorized or the policy server cannot be contacted.

This action can succeed when a configuration file does not exist. When the configuration file does not exist, it is created and used as a temporary file to hold configuration information during the operation, and then the file is deleted completely.

If this action is specified, the following options are required: `-admin_id`, `-admin_pwd`, `-appsvr_id`, and `-policysvr`.

-admin_id *admin_user_ID*
Specifies the Tivoli Access Manager administrator name. If this option is not specified, `sec_master` is the default.

A valid administrative ID is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the administrative ID.

For example, for U.S. English the valid characters are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). The minimum and maximum lengths of the administrative ID, if there are limits, are imposed by the underlying registry.

-admin_password *admin_password*

Specifies the password of the Tivoli Access Manager administrator user that is associated with the **-admin_id** parameter. The password restrictions depend upon the password policy for your Tivoli Access Manager configuration.

-appsvr_id *application_server_name*
Specifies the name of the application server. The name is combined with the host name to create unique names for Tivoli Access Manager objects created for your application. The following names are reserved for Tivoli Access Manager applications: ivacld, secmgrd, ivnet, and ivweb.

-appsvr_pwd *application_server_password*
Specifies the password of the application server. This option is required. A password is created by the system and the configuration file is updated with the password created by the system.

If this option is not specified, the server password will be read from standard input.

-authzsvr *authorization_server_name*
Specifies the name of the Tivoli Access Manager authorization server with which the application server communicates. The server is specified by fully qualified host name, the SSL port number, and the rank. The default SSL port number is 7136. For example: myauth.mycompany.com:7136:1. You can specify multiple servers if the entries are separated by a comma (,).

-cfg_action {create | replace}
Specifies the action to take when creating the configuration and key files. Valid values are **create** or **replace**. Use the **create** option to initially create the configuration and keystore files. Use the **replace** option if these files already exist. If you use the **create** option and the configuration or keystore files already exist, an exception is created.

Options are as follows:

create Specifies to create the configuration and key store files during server configuration. Configuration fails if either of these files already exists.

replace
Specifies to replace the configuration and key store files during server configuration. Configuration deletes any existing files and replaces them with new ones.

-domain *Tivoli_Access_Manager_domain*
Specifies the Tivoli Access Manager domain name to which the administrator is authenticated. This domain must exist and an the administrator ID and password must be valid for this domain. The application server is specified in this domain.

If not specified, the local domain that was specified during Tivoli Access Manager runtime configuration will be used. The local domain value will be retrieved from the configuration file.

A valid domain name is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the domain name.

For example, for U.S. English the valid characters for domain names are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). The minimum and maximum lengths of the domain name, if there are limits, are imposed by the underlying registry.

-key_file *fully_qualified_name_of_keystore_file*
Specifies the directory that is to contain the key files for the server. A valid directory name is determined by the operating system. Use a fully qualified file name that contains the application server certificate and key file.

Make sure that server user (for example, `ivmgr`) or all users have permission to access the `.kdb` file and the folder that contains the `.kdb` file.

This option is required.

-mode *server_mode*
Specifies the mode in which the application server processes requests. Only the `remote` mode is supported.

-policysvr *policy_server_name*
Specifies the names of servers that run the Tivoli Access Manager policy server (ivmgrd) with which the application server communicates. A server is specified by a fully qualified host name, the SSL port number, and the rank. The default SSL port number is 7135. For example: `mypolicy.mycompany.com:7135:1`. You can specify multiple servers if the entries are separated by a comma (,).

-port *port_number*
Specifies the TCP/IP communications port on which the application server listens for communications from the policy servers.

-profileName *profile_name*
Specifies the name of your WebSphere Application Server profile. If this option is not specified, the default `server1` profile is used.

Creating a trusted user account in Tivoli Access Manager:

Tivoli Access Manager trust association interceptors require the creation of a trusted user account in the shared LDAP user registry.

About this task

This account includes the ID and password that WebSEAL uses to identify itself to WebSphere Application Server. To prevent potential vulnerabilities, do not use the sec_master ID as the trusted user account and ensure that the password you use is unique and generated randomly. Use the trusted user account for the TAI or TAI++ only.

Procedure

1. Use either the Tivoli Access Manager pdadmin command-line utility or Web Portal Manager to create the trusted user. For example, from the **pdadmin** command line.
2. Reference the code listed below as an example for creating a trusted user account.
3. Reference the following additional resources for more information:
   a. “Configuring WebSEAL for use with WebSphere Application Server”
   b. “Configuring Tivoli Access Manager plug-in for web servers for use with WebSphere Application Server” on page 1508

Example

```
pdadmin> user create webseal_userid webseal_userid_DN firstname
         surname password

pdadmin> user modify webseal_userid account-valid yes
```

Configuring WebSEAL for use with WebSphere Application Server:

Use this topic to set the SSO password in WebSEAL for single sign-on to WebSphere Application Server.

About this task

A junction must be created between WebSEAL and WebSphere Application Server. This junction carries the iv-credentials (for TAI++) or iv-user (for TAI) and the HTTP basic authentication headers with the request. You can configure WebSEAL to pass the end user identity in other ways, the iv-credentials header is the only one supported by the TAI++ and the iv-user is the only one supported by TAI.

Communications over the junction should use Secure Sockets Layer (SSL) for increased security. Setting up SSL across this junction requires that you configure the HTTP Server used by WebSphere Application Server, and WebSphere Application Server itself, to accept inbound SSL traffic and route it correctly to WebSphere Application Server. This activity requires importing the necessary signing certificates into the WebSEAL certificate keystore, and possibly also the HTTP Server certificate keystore.

Create the junction between WebSEAL and WebSphere Application Server using the **-c iv_creds** option for TAI++ and **-c iv_user** for TAI. Enter either of the following commands as one line using the variables that are appropriate for your environment:

TAI++

```
server task webseald-server create -t ssl -b supply -c iv_creds
 -h host_name -p websphere_app_port_number junction_name
```

TAI

```
server task webseald-server create -t ssl -b supply -c iv_user
 -h host_name -p websphere_app_port_number junction_name
```

Notes:

1. If warning messages are displayed about the incorrect setup of certificates and key databases, delete the junction, correct problems with the key databases, and recreate the junction.
2. The junction can be created as `-t tcp` or `-t ssl`, depending on your requirements.

For single sign-on (SSO) to WebSphere Application Server the SS) password must be set in WebSEAL. To set the password, complete the following steps:

Procedure

1. Edit the WebSEAL configuration file *webseal_install_directory*`/etc/webseald-default.conf` Set the following parameter: `basicauth-dummy-passwd=`*webseal_userid_passwd*

   where *webseal_userid_passwd* is the SSO password for the trusted user account set in “Creating a trusted user account in Tivoli Access Manager” on page 1507.
2. Restart WebSEAL.

What to do next

For more details and options about how to configure junctions between WebSEAL and WebSphere Application Server, including other options for specifying the WebSEAL server identity, refer to the *Tivoli Access Manager WebSEAL Administration Guide* as well as to the documentation for the HTTP Server you are using with your WebSphere Application Server. Tivoli Access Manager documentation is available at http://publib.boulder.ibm.com/tividd/td/tdprodlist.html.

Configuring Tivoli Access Manager plug-in for web servers for use with WebSphere Application Server:

Tivoli Access Manager plug-in for web servers can be used as a security gateway for your protected WebSphere Application Server resources.

About this task

With such an arrangement the plug-in authorizes all user requests before passing the credentials of the authorized user to WebSphere Application Server in the form of an iv-creds header. Trust between the plug-in and WebSphere Application Server is established through use of basic authentication headers containing the single sign-on (SSO) user password.

Procedure

1. The Tivoli Access Manager plug-in for web servers configuration shows IV headers configured for post-authorization processing, and basic authentication that is configured as the authentication mechanism and for post-authorization processing, as shown in the example below.
2. After a request is authorized, the basic authentication header is removed from the request (`strip-hdr=always`) and a new one is added (`add-hdr=supply`).
3. Included in this new header is the password that is set when the SSO user is created in "Creating a trusted user account in Tivoli Access Manager" on page 1507.
4. Specify this password in the **supply-password** parameter and it is passed in the newly created header. This basic authentication header enables trust between WebSphere Application Server and the plug-in.
5. An iv-creds header is also added (`generate=iv-creds`), which contains the credential information of the user passed onto WebSphere Application Server. Session cookies are used to maintain session state.

Example

```
[common-modules]
authentication = BA
session = session-cookie
post-authzn = BA
post-authzn = iv-headers

[iv-headers]
accept = all
generate = iv-creds

[BA]
strip-hdr = always
add-hdr = supply
supply-password = sso_user_password
```

What to do next

"Configuring single sign-on using trust association" or "Configuring single sign-on using trust association interceptor ++" on page 1510

Configuring single sign-on using trust association:

This task is performed to enable single sign-on using trust association. Trust association is used to connect reversed proxy servers to the application server.

Before you begin

Note: Use of TAIs for Simple and Protected GSS-API Negotiation Mechanism (SPNEGO) authentication is deprecated in this release. The SPNEGO web authentication panels provide a much easier and less error-prone way to configure SPNEGO.

To establish the trust association for the single sign-on, perform the following steps:

Procedure

1. From the administrative console for WebSphere Application Server, click **Security > Global security**.
2. From Authentication mechanisms, click **Web and SIP security > Trust association**.
3. Select the **Enable trust association** option.

4. Under Additional properties, click the **Interceptors** link.
5. Click **com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus** to use a WebSEAL interceptor, or **com.ibm.ws.security.spnego.TrustAssociationInterceptorImpl** to use a SPNEGO interceptor.
6. Under Custom properties, select a custom property to edit or click **New** to create a new one. Enter the property name and value pairs.
7. Click **OK**.
8. Save the configuration and log out.
9. Restart WebSphere Application Server.

Configuring single sign-on using trust association interceptor ++:

Perform this task to enable single sign-on using trust association interceptor ++. The steps involve setting up trust association and creating the interceptor properties.

Before you begin

Lightweight Third Party Authentication (LTPA) is the default authentication mechanism for WebSphere Application Server. However, you may need to configure LTPA prior to configuring the TAMTrustAssociationInterceptorPlus. LTPA is the required authentication mechanism for all trust association interceptors. You can configure LTPA by clicking **Security > Global security > Authentication mechanisms and expiration**.

Note: Enabling web security single sign-on (SSO) is optional when you configure the TAMTrustAssociationInterceptorPlus. For more information, see “Implementing single sign-on to minimize web user authentications” on page 1441.

Although you can use Simple WebSphere Authentication Mechanism (SWAM) by selecting the **Use SWAM-no authenticated communication between servers** option on the Authentication mechanisms and expiration panel, single sign-on (SSO) requires LTPA as the configured authentication mechanism.

To establish the trust association for the single sign-on, perform the following steps:

Procedure

1. From the administrative console for WebSphere Application Server, click **Security > Global security**.
2. Under Web security, click **Trust association**.
3. Click **Enable Trust Association**.
4. Click **Interceptors**.
5. Click **com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus** to use a WebSEAL interceptor. This interceptor is one of two WebSEAL interceptors that are supplied for your use. You choose to use this interceptor by supplying properties as described in the next step.

   Attention: WebSphere Application Server attempts to initialize both of these interceptors even if you only supplied properties for the **com.ibm.ws.security.web.TAMTrustAssociationInterceptorPlus** interceptor. As a result, messages AWXRB0008E and SECJ0384E can appear during initialization to indicate that the interceptor you did not choose has failed to initialize. This is normal processing and does not affect the initialization of the interceptor you did select. To inhibit the display of messages AWXRB0008E and SECJ0384E, you can delete the interceptor you do not want to use prior to beginning the initialization. You can add that interceptor back later if your environment changes.
6. Click **Custom Properties**.
7. Click **New** to enter the property name and value pairs. Ensure that the following parameters are set:

Table 100. Custom properties.

This table describes the TAI custom properties.

Option	Description
com.ibm.websphere.security.webseal.checkViaHeader	You can configure TAI so that the via header can be ignored when validating trust for a request. Set this property to *false* if none of the hosts in the via header need to be trusted. When set to *false* you do not need to set the trusted host names and host ports properties. The only mandatory property to check when via header is *false* is com.ibm.websphere.security.webseal.loginId. The default value of the check via header property is *false*. When using Tivoli Access Manager plug-in for web servers, set this property to *false*. **Note:** The via header is part of the standard HTTP header that records the server names the request that passed through.
com.ibm.websphere.security.webseal.loginId	The WebSEAL trusted user as created in "Creating a trusted user account in Tivoli Access Manager" on page 1507 The format of the username is the short name representation. This property is mandatory. If it is not set in WebSphere Application Server, the TAI initialization fails.
com.ibm.websphere.security.webseal.id	A comma-separated list of headers that exists in the request. If all of the configured headers do not exist in the request, trust cannot be established. The default value for the ID property is *iv-creds*. Any other values set in WebSphere Application Server are added to the list along with iv-creds, separated by commas.
com.ibm.websphere.security.webseal.hostnames	Do not set this property if using Tivoli Access Manager Plug-in for Web Servers. The property specifies the host names (case sensitive) that are trusted and expected in the request header. Requests arriving from un-listed hosts might not be trusted. If the checkViaHeader property is not set or is set to false then the trusted host names property has no influence. If the checkViaHeader property is set to `true`, and the trusted host names property is not set, TAI initialization fails.
com.ibm.websphere.security.webseal.ports	Do not set this property if using Tivoli Access Manager plug-in for web servers. This property is a comma-separated list of trusted host ports. Requests that arrive from unlisted ports might not be trusted. If the checkViaHeader property is not set, or is set to `false` this property has no influence. If the checkViaHeader property is set to `true`, and the trusted host ports property is not set in WebSphere Application Server, the TAI initialization fails.

com.ibm.websphere.security.webseal.viaDepth	A positive integer that specifies the number of source hosts in the via header to check for trust. By default, every host in the via header is checked, and if any host is not trusted, trust cannot be established. The via depth property is used when only some of the hosts in the via header have to be trusted. The setting indicates the number of hosts that are required to be trusted.
	
	As an example, consider the following header:
	`Via: HTTP/1.1 webseal1:7002, 1.1 webseal2:7001`
	
	If the viaDepth property is not set, is set to 2 or is set to 0, and a request with the previous via header is received then both `webseal1:7002` and `webseal2:7001` need to be trusted. The following configuration applies:
	`com.ibm.websphere.security.webseal.hostnames = webseal1,webseal2`
	`com.ibm.websphere.security.webseal.ports = 7002,7001`
	
	If the via depth property is set to 1, and the previous request is received, then only the last host in the via header needs to be trusted. The following configuration applies:
	`com.ibm.websphere.security.webseal.hostnames = webseal2`
	`com.ibm.websphere.security.webseal.ports = 7001`
	
	The viaDepth property is set to `0` by default, which means all of the hosts in the via header are checked for trust.

com.ibm.websphere.security.webseal.ssoPwdExpiry	After trust is established for a request, the single sign-on user password is cached, eliminating the need to have the TAI re-authenticate the single sign-on user with Tivoli Access Manager for every request. You can modify the cache timeout period by setting the single sign-on password expiry property to the required time in seconds. If the password expiry property is set to `0`, the cached password never expires. The default value for the password expiry property is `600`.
com.ibm.websphere.security.webseal.ignoreProxy	This property can be used to tell the TAI to ignore proxies as trusted hosts. If set to `true` the comments field of the hosts entry in the via header is checked to determine if a host is a proxy. Remember that not all proxies insert comments in the via header indicating that they are proxies. The default value of the ignoreProxy property is `false`. If the checkViaHeader property is set to `false` then the ignoreProxy property has no influence in establishing trust.
com.ibm.websphere.security.webseal.configURL	Set this property to *profile_root*`/etc/pd/PolicyDirector/PDPerm.properties`. For the TAI to establish trust for a request, it requires that a PDPerm.properties file exists in each node within the cell. Also, the correct URL of the properties file must be set in the config URL property. If this property is not set or the `PDPerm.properties` file is not in the specified location, the TAI initialization fails. The `PDPerm.properties` file is part of the Tivoli Access Manager configuration for a node. To create the Tivoli Access Manager configuration, run the **pdjrtecfg** script and then the **svrsslcfg** script for each node in the cell. The `PDPerm.properties` file is created in the*profile_root*`/etc/pd/PolicyDirector/` directory.

8. Click **OK**.

9. Save the configuration and log out.
10. Restart WebSphere Application Server.

Configuring global sign-on principal mapping:

You can create a new application login that uses the Tivoli Access Manager GSO database to store the login credentials.

Procedure

1. Click **Security > Global security**.
2. Under Authentication, click **Java Authentication and Authorization Service > Application logins**.
3. Click **New** to create a new Java Authentication and Authorization Service (JAAS) login configuration.
4. Enter the alias name of the new application login. Click **Apply**.
5. Under Additional properties, click **JAAS login modules** to define the JAAS Login Modules.
6. Click **New** and enter the following information:

   Module class name: com.tivoli.pdwas.gso.AMPrincipalMapper
   Use Login Module Proxy: enable
   Authentication strategy: REQUIRED
7. Click **Apply**
8. Under Additional Properties section, click **Custom Properties** to define login module-specific values that are passed directly to the underlying login modules.
9. Click **New**.

   The Tivoli Access Manager principal mapping module uses the `authDataAlias` configuration string to retrieve the correct user name and password from the security configuration.

   The authDataAlias attribute that is passed to the module is configured for the J2C connection factory. Because the authDataAlias attribute is an arbitrary string that is entered at configuration time, the following scenarios are possible:

   - The authDataAlias attribute contains both the global sign-on (GSO) resource name and the user name. The format of this string is `"Resource/User"`.
   - The authDataAlias attribute contains the GSO Resource name only. The user name is determined by using the Subject of the current session.

   The scenario to use is determined by a JAAS configuration option, as shown here:

   Name: com.tivoli.pd.as.gso.AliasContainsUserName

   Value: True, if the alias contains the user name; false, if the user name must be retrieved from the security context

   When entering `authDataAlias` attributes through the WebSphere Application Server administrative console, the node name is automatically pre-pended to the alias. The JAAS configuration entry determines whether this node name is removed or included as part of the resource name, as shown here:

   Name: com.tivoli.pd.as.gso.AliasContainsNodeName

   Value: True, if the alias contains the node name

   Note: If the `PdPerm.properties` configuration file is not located in the `JAVA_HOME/PdPerm.properties` default location, then you also need to add the following property:

   Name: com.tivoli.pd.as.gso.AMCfgURL

   Value: file:///*path to PdPerm.properties*

   Enter each new parameter using the following scenario information as a guide, then click **Apply**.

Scenario 1
Auth Data Alias - BackendEIS/eisUser
Resource - BackEndEIS
User - eisUser
Principal Mapping Parameters

Table 101. Principal Mapping Parameters.

This table lists the principal mapping parameters.

Name	Value
delegate	com.tivoli.pdwas.gso.AMPrincipalMapper
com.tivoli.pd.as.gso.AliasContainsUserName	true
com.tivoli.pd.as.gso.AliasContainsNodeName	false
com.tivoli.pd.as.gso.AMLoggingURL	file:///*jlog_props_path*
debug	false

Scenario 2
Auth Data Alias - BackendEIS
Resource - BackEndEIS
User - Currently authenticated WebSphere Application Server user
Principal Mapping Parameters

Table 102. Principal Mapping Parameters.

This table lists the principal mapping parameters.

Name	Value
delegate	com.tivoli.pdwas.gso.AMPrincipalMapper
com.tivoli.pd.as.gso.AliasContainsUserName	false
com.tivoli.pd.as.gso.AliasContainsNodeName	false
com.tivoli.pd.as.gso.AMLoggingURL	file:///*jlog_props_path*
debug	false

Scenario 3
Auth Data Alias - nodename/BackendEIS/eisUser
Resource - BackEndEIS
User - eisUser
Principal Mapping Parameters

Table 103. Principal Mapping Parameters.

This table lists the principal mapping parameters.

Name	Value
delegate	com.tivoli.pdwas.gso.AMPrincipalMapper
com.tivoli.pd.as.gso.AliasContainsUserName	true
com.tivoli.pd.as.gso.AliasContainsNodeName	true
com.tivoli.pd.as.gso.AMLoggingURL	file:///*jlog_props_path*
debug	false

Scenario 4
Auth Data Alias - nodename/BackendEIS/eisUser
Resource - nodename/BackEndEIS (notice that node name is not removed)
User - eisUser
Principal Mapping Parameters

Table 104. Principal Mapping Parameters.

This table lists the principal mapping parameters.

Name	Value
delegate	com.tivoli.pdwas.gso.AMPrincipalMapper
com.tivoli.pd.as.gso.AliasContainsUserName	true
com.tivoli.pd.as.gso.AliasContainsNodeName	false
com.tivoli.pd.as.gso.AMLoggingURL	file:///*jlog_props_path*
debug	false

Scenario 5
Auth Data Alias - BackendEIS/eisUser
Resource - BackEndEIS
User - eisUser
Principal Mapping Parameters

Table 105. Principal Mapping Parameters.

This table lists the principal mapping parameters.

Name	Value
delegate	com.tivoli.pdwas.gso.AMPrincipalMapper
com.tivoli.pd.as.gso.AliasContainsUserName	false
com.tivoli.pd.as.gso.AliasContainsNodeName	true
com.tivoli.pd.as.gso.AMLoggingURL	file:///*jlog_props_path*
debug	false

Scenario 6
Auth Data Alias - nodename/BackendEIS/eisUser
Resource - nodename/BackendEIS/eisUser
(notice that the resource is the same as Auth Data Alias).
User - Currently authenticated WebSphere Application Server user
Principal Mapping Parameters

Table 106. Principal Mapping Parameters.

This table lists the principal mapping parameters.

Name	Value
delegate	com.tivoli.pdwas.gso.AMPrincipalMapper
com.tivoli.pd.as.gso.AliasContainsUserName	false
com.tivoli.pd.as.gso.AliasContainsNodeName	false
com.tivoli.pd.as.gso.AMLoggingURL	file:///*jlog_props_path*
debug	false

10. Create the Java 2 Connector (J2C) authentication aliases. The user name and password that are assigned to these alias entries are irrelevant because Tivoli Access Manager is responsible for providing user names and passwords. However, the user name and password that are assigned to the J2C authentication aliases need to exist so that they can be selected for the J2C connection factory in the administrative console.

    To create the J2C authentication aliases, from the WebSphere Application Server administrative console, click **Security** >**Global security**. Under Authentication, click **Java Authentication and Authorization Service** > **J2C authentication data**, and then click **New** for each new entry. Refer to the previous table for scenario inputs.

    The connection factories for each resource adapter that need to use the GSO database must be configured to use the Tivoli Access Manager Principal mapping module:

a. From the WebSphere Application Server administrative console, click **Applications > Enterprise Applications** > *application_name* **> Resourcer references**. Note that J2C connection factories must be already configured for the selected application. To configure a new J2C connection factory, see the Configuring Java EE Connector connection factories in the administrative console article.

b. Under Additional properties, click **Resource Adapter**.

   The resource adapter can be stand-alone and does not need to be packaged with the application. The resource adapter is configured from **Resources** > **Resource Adapters** for stand-alone scenarios.

c. Under Additional properties, click **J2C Connection Factories**.

d. Click **New** and enter the connection factory properties.

e. When finished, click **Apply > Save**.

Attention:
Custom mapping configuration for the connection factory is deprecated in WebSphere Application Server Version 6. To configure the GSO credential mapping, use the Map Resource References to Resources panel on the administrative console. For more information, see the J2EE connector security article.

Configuring administrative authentication

An authentication mechanism defines rules about security information, such as whether a credential is forwardable to another Java process, and the format of how security information is stored in both credentials and tokens. The Rivest Shamir Adleman (RSA) token authentication mechanism simplifies the security environment for flexible management topology, that is, the topology where you can locally or remotely submit and manage administrative jobs through a job manager that manages applications, perform product maintenance, modify configurations, and control the application server runtime. You use the administrative console to configure administrative authentication, which involves the configuring of the Rivest Shamir Adleman (RSA) token authentication mechanism.

Before you begin

The following keystore, truststore, and rootstore descriptions give you an idea of where certificates are stored and how trust is configured between processes.

The **NodeRSATokenKeyStore** contains the Rivest Shamir Adleman (RSA) token personal certificate used for this process. Not only is the public/private key from this certificate used to create RSA tokens, but the public key is used by other processes to create tokens. The RSA personal certificate is signed by an RSA root certificate.

The **NodeRSATokenTrustStore** contains all RSA signer certificates from other processes that are trusted to send RSA tokens to this process. The signers in this trust store are placed there automatically during the registration process. However, this task allows an administrator to configure trust between to processes not normally involved in the same administrative domain. There may be requirements where two base servers are communicating administratively. When using the RSA token authentication mechanism, the base servers need to share RSA signers if administrative communications is operating in both directions.

The NodeRSATokenRootStore contains the root personal certificate that is used to create new RSA personal certificates. Do not use the root certificate to create RSA tokens because this usage compromises the long-lived keys. Only use the root certificate to sign other certificates.

No manual steps are required with these keystores, and this allows uncommon trust establishment among processes not in the same administrative domain. You can also replace the RSA personal certificate with a personal certificate obtained from a certificate authority (CA) if desired. In this case, make sure the CA root

certificate is placed in all RSA trust stores in the same administrative domain.

Procedure

1. Click **Security** > **SSL certificate and key management**.
2. Under **Related Items**, click **Key stores and certificates**.
3. Under **Keystore usages**, select **RSA token keystores**.
4. Select the RSA token key store you want to administer.
5. Modify the description if required.
6. Modify the path if required.
7. Select **read only**, **initialize at setup**, or both if required.
8. Enter the correct password to make these modifications
9. Click **Apply** and **Save**.

Results

You configured administrative authentication.

What to do next

In cases where the process is back-level or a target RSA certificate cannot be obtained, the fallback mechanism is Lightweight Third-Party Authentication (LTPA) which is supported in all previous releases for administrative communications. The fallback occurs automatically. If the LTPA keys are not shared and a fallback occurs, LTPA will fail as well. However, this situation is typically an error case in the RSA mechanism and should occur infrequently.

Java Authentication and Authorization Service

The standard Java 2 security application programming interface (API) helps enforce access control based on the location of the code source or the author or packager of the code that signed the jar file. The current principal of the running thread is not considered in the Java 2 security authorization. Instances where authorization is based on the principal, as opposed to the code base, and the user exist. The Java Authentication and Authorization Service is a standard Java API that supports the Java 2 security authorization to extend the code base on the principal as well as the code base and users.

The Java Authentication and Authorization Service (JAAS) Version 1.0 extends the Java 2 security architecture of the Java 2 platform with additional support to authenticate and enforce access control with principals and users. JAAS implements a Java version of the standard Pluggable Authentication Module (PAM) framework, and extends the access control architecture of the Java 2 platform in a compatible fashion to support user-based authorization or principal-based authorization. WebSphere Application Server fully supports the JAAS architecture. JAAS extends the access control architecture to support role-based authorization for Java Platform, Enterprise Edition (Java EE) resources including servlets, JavaServer Pages (JSP) files, and Enterprise JavaBeans (EJB) components.

Refer to “Java 2 security” on page 1176 for more information.

The following sections cover the JAAS implementation and programming model:
- Login configuration for Java Authentication and Authorization Service
- Programmatic login for JAAS
- “Java Authentication and Authorization Service authorization” on page 1517

The JAAS documentation can be found at http://www.ibm.com/developerworks/java/jdk/security. Scroll down to find the JAAS documentation for your platform.

Java Authentication and Authorization Service authorization

Java 2 security architecture uses a security policy to specify which access rights are granted to running code. This architecture is *code-centric*. The permissions are granted based on code characteristics including where the code is coming from, whether it is digitally signed, and by whom. Authorization of the Java Authentication and Authorization Service (JAAS) augments the existing code-centric access controls with new user-centric access controls. Permissions are granted based on what code is running and who is running it.

When using JAAS authentication to authenticate a user, a *subject* is created to represent the authenticated user. A subject is comprised of a set of principals, where each principal represents an identity for that user. You can grant permissions in the policy to specific principals. After the user is authenticated, the application can associate the subject with the current access control context. For each subsequent security-checked operation, the Java runtime automatically determines whether the policy grants the required permission to a specific principal only. If so, the operation is supported if the subject that is associated with the access control context contains the designated principal only.

Associate a subject with the current access control context by calling the static doAs method from the subject class, passing it an authenticated subject and the java.security.PrivilegedAction or java.security.PrivilegedExceptionAction method. The doAs method associates the provided subject with the current access control context and then invokes the run method from the action. The run method implementation contains all the code that ran as the specified subject. The action runs as the specified subject.

In the Java 2 Platform, Enterprise Edition (J2EE) programming model, when invoking the Enterprise JavaBeans (EJB) method from an enterprise bean or servlet, the method runs under the user identity that is determined by the run-as setting. The J2EE Version 1.4 Specification does not indicate which user identity to use when invoking an enterprise bean from a Subject.doAs action block within either the EJB code or the servlet code. A logical extension is to use the proper identity that is specified in the subject when invoking the EJB method within the Subject doAs action block.

Letting the Subject.doAs action overwrite the run-as identity setting is an ideal way to integrate the JAAS programming model with the J2EE run-time environment. However, JAAS introduced an issue into the Software Development Kit (SDK), Java Technology Edition Versions 1.3 or later when integrating the JAAS Version 1.0 or later implementation with the Java 2 security architecture. A subject, which is associated with the access control context is cut off by a doPrivileged call when a doPrivileged call occurs within the Subject.doAs action block. Until this problem is corrected, no reliable and run-time efficient way is available to guarantee the correct behavior of Subject.doAs action in a J2EE run-time environment.

The problem can be explained better with the following example:

```
Subject.doAs(subject, new java.security.PrivilegedAction() {
   Public Object run() {
       // Subject is associated with the current thread context
       java.security.AccessController.doPrivileged( new
           java.security.PrivilegedAction() {
                        public Object run() {
                        // Subject was cut off from the current
                        // thread context
   return null;
                }
        });
        // Subject is associated with the current thread context
        return null;
   }
});
```

In the previous code example, the Subject object is associated with the context of the current thread. Within the run method of a doPrivileged action block, the Subject object is removed from the thread context. After leaving the doPrivileged block, the Subject object is restored to the current thread context. Because doPrivileged blocks can be placed anywhere along the running path and instrumented quite often in a server environment, the run-time behavior of a doAs action block becomes difficult to manage.

To resolve this difficulty, WebSphere Application Server provides a WSSubject helper class to extend the JAAS authorization to a J2EE EJB method invocation, as described previously. The WSSubject class provides static doAs and doAsPrivileged methods that have identical signatures to the subject class. The WSSubject.doAs method associates the Subject to the currently running thread. The WSSubject.doAs and WSSubject.doAsPrivileged methods then invoke the corresponding Subject.doAs and Subject.doAsPrivileged methods. The original credential is restored and associated with the running thread upon leaving the WSSubject.doAs and WSSubject.doAsPrivileged methods.

The WSSubject class is not a replacement of the subject object, but rather a helper class to ensure consistent run-time behavior as long as an EJB method invocation is a concern.

The following example illustrates the run-time behavior of the WSSubject.doAs method:

```
WSSubject.doAs(subject, new java.security.PrivilegedAction() {
   Public Object run() {
       // Subject is associated with the current thread context
       java.security.AccessController.doPrivileged( new
           java.security.PrivilegedAction() {
                      public Object run() {
                         // Subject was cut off from the current thread
                         // context.
   return null;
              }
        });
        // Subject is associated with the current thread context
   return null;
   }
});
```

The Subject.doAs and Subject.doAsPrivileged methods are not integrated with the J2EE run-time environment. EJB methods that are invoked within the Subject.doAs and Subject.doAsPrivileged action blocks run under the identity that is specified by the run-as setting and not by the subject identity.

- The Subject object that is generated by the WSLoginModuleImpl instance and the WSClientLoginModuleImpl instance contains a principal that implements the WSPrincipal interface. Using the getCredential method for a WSPrincipal object returns an object that implements the WSCredential interface. You can also find the WSCredential object instance in the PublicCredentials list of the subject instance. Retrieve the WSCredential object from the PublicCredentials list instead of using the getCredential method.
- The getCallerPrincipal method for the WSSubject class returns a string that represents the caller security identity. The return type differs from the getCallerPrincipal method of the java.security.Principal EJBContext interface.
- The Subject object that is generated by the Java 2 Connector (J2C) DefaultPrincipalMapping module contains a resource principal and a PasswordCredentials list. The resource principal represents the RunAs identity.

For more information, see J2EE connector security.

Using the Java Authentication and Authorization Service programming model for web authentication

WebSphere Application Server supports the Java Platform, Enterprise Edition (Java EE) declarative security model. You can define the authentication and access control policy using the Java EE deployment descriptor. You can further stack custom login modules to customize the WebSphere Application Server authentication mechanism.

Before you begin

A custom login module can perform principal and credential mapping, custom security token and custom credential-processing, and error-handling among other possibilities. Typically, you do not need to use application code to perform authentication function. Use the programming techniques that are described in this section if you have to perform authentication function in application code. For example, if you have applications that programmed to the SSOAuthenticator helper function, you can use the following programming interface. The SSOAuthenticator helper function was deprecated starting with WebSphere Application Server Version 4.0. Use declarative security as a rule; use the techniques that are described in this section as a last resort.

About this task

When the Lightweight Third-Party Authentication (LTPA) mechanism single sign-on (SSO) option is enabled, the web client login session is tracked by an LTPA SSO token cookie after successful login. At logout, this token is deleted to terminate the login session, but the server-side subject is not deleted. When you use the declarative security model, the WebSphere Application Server web container performs client authentication and login session management automatically. You can perform authentication in application code by setting a login page without a Java EE security constraint and by directing client requests to your login page first. Your login page can use the Java Authentication and Authorization Service (JAAS) programming model to perform authentication. To enable WebSphere Application Server web login modules to generate SSO cookies, use the following steps.

Procedure

1. Create a new system login JAAS configuration. To access the panel, click **Security > Global security**. Under Java Authentication and Authorization Service, click **System logins**.
2. Manually clone the WEB_INBOUND login configuration, and give it a new alias. To clone the login configuration, click **New**, enter a name for the configuration, click **Apply**, then click **JAAS login modules** under Additional properties. Click **New** and configure the JAAS login module. For more information, see Login module settings for Java Authentication and Authorization Service. WebSphere Application Server web container uses the WEB_INBOUND login configuration to authenticate web clients. Changing the WEB_INBOUND login configuration affects all web applications in the cell. You should create your own login configuration by cloning the contents of the WEB_INBOUND login configuration.
3. Select the `wsMapDefaultInboundLoginModule` login module and click **Custom properties**. There are two login modules defined in your login configuration: `ltpaLoginModule` and `wsMapDefaultInboundLoginModule`.
4. Add a login property name *cookie* with a value of **true**. The two login modules are enabled to generate LTPA SSO cookies. Do not add the cookie login option to the original WEB_INBOUND login configuration.
5. Stack your custom LoginModule(s) in the new login configuration (optional).
6. Use your login page for programmatic login by perform a JAAS LoginContext.login using your newly defined login configuration. After a successful login, either the `ltpaLoginModule` or the `wsMapDefaultInboundLoginModule` generates an LTPA SSO cookie upon a successful authentication. Exactly which LoginModule generates the SSO cookie depends on many factors, including system authentication configuration and runtime condition (which is beyond the scope of this section).

7. Call the modified `WSSubject.setRunAsSubject` method to add the subject to the authentication cache. The subject must be a WebSphere Application Server JAAS subject created by `LoginModule`. Adding the subject to the authentication cache recreates a subject from SSO token.
8. Use your programmatic logout page to revoke SSO cookies by invoking the `revokeSSOCookies` method from the WSSecurityHelper class. The term cookies is used because WebSphere Application Server Release 5.1.1 (and later) release supports a new LTPA SSO token with a different encryption algorithm, but can be configured to generate the original LTPA SSO token for backward compatibility. Note that the subject is still in the authentication cache and only the SSO cookies are revoked.

Example

Use the following code sample to perform authentication.

gotcha: If you set the password for the WSCallbackHandlerFactoryset factory class for getting handlers to `null`, as is done in the following example, you allow identity assertion without a password.

```
Suppose you wrote a LoginServlet.java:

 Import com.ibm.wsspi.security.auth.callback.WSCallbackHandlerFactory;
 Import com.ibm.websphere.security.auth.WSSubject;

 public Object login(HttpServletRequest req, HttpServletResponse res)
 throws ServletException {

 PrintWriter out = null;
 try {
  out =  res.getWriter();
      res.setContentType("text/html");
 } catch (java.io.IOException e){
  // Error handling
 }

 Subject subject = null;
 try {
 LoginContext lc = new LoginContext("system.Your_login_configuration",
WSCallbackHandlerFactory.getInstance().getCallbackHandler(
userid, null, password, req, res, null));
  lc.login();
  subject = lc.getSubject();
      WSSubject.setRunAsSubject(subject);
 } catch(Exception e) {
  // catch all possible exceptions if you want or handle them separately
  out.println("Exception in LoginContext login + Exception = " +
 e.getMessage());
  throw new ServletException(e.getMessage());
 }

The following is sample code to revoke the SSO cookies upon a programming logout:


The LogoutServlet.java:

 public void logout(HttpServletRequest req, HttpServletResponse res,
 Object retCreds)  throws ServletException {
   PrintWriter out =null;
   try {
    out =  res.getWriter();
        res.setContentType("text/html");
   } catch (java.io.IOException e){
   // Error Handling
   }
   try {
     WSSecurityHelper.revokeSSOCookies(req, res);
   } catch(Exception e) {
    // catch all possible exceptions if you want or handle them separately
    out.println("JAASLogoutServlet: logout Exception = " + e.getMessage());
    throw new ServletException(e);
   }
  }
```

What to do next

For more information on JAAS authentication, refer to Developing programmatic logins with the Java Authentication and Authorization Service. For more information on the AuthenLoginModule login module, refer to Example: Customizing a server-side Java Authentication and Authorization Service authentication and login configuration.

Developing custom login modules for a system login configuration for JAAS

For WebSphere Application Server, multiple Java Authentication and Authorization Service (JAAS) plug-in points exist for configuring system logins. WebSphere Application Server uses system login configurations to authenticate incoming requests, outgoing requests, and internal server logins.

About this task

Application login configurations are called by Java Platform, Enterprise Edition (Java EE) applications for obtaining a Subject that is based on specific authentication information. This login configuration enables the application to associate the Subject with a specific protected remote action. The Subject is picked up on the outbound request processing. The following list identifies the main system plug-in points. If you write a login module that adds information to the Subject of a system login, these are the main login configurations to plug in:

- WEB_INBOUND
- RMI_OUTBOUND
- RMI_INBOUND
- DEFAULT

Procedure

- Authenticate web requests with the WEB_INBOUND login configuration.

  The WEB_INBOUND login configuration authenticates web requests.

  For more detailed information on the WEB_INBOUND configuration including its associated callbacks, see "RMI_INBOUND, WEB_INBOUND, DEFAULT" in System login configuration entry settings for Java Authentication and Authorization Service. Figure 1 shows an example of a configuration using a trust association interceptor (TAI) that creates a Subject with the initial information that is passed into the WEB_INBOUND login configuration. If the trust association interceptor is not configured, the authentication process goes directly to the WEB_INBOUND system login configuration, which consists of all the login modules combined in Figure 1. Figure 1 shows where you can plug in custom login modules and where the ltpaLoginModule and the wsMapDefaultInboundLoginModule login modules are required.

  Figure 1

- Handle outbound requests with the RMI_OUTBOUND login configuration.

  The RMI_OUTBOUND login configuration is a plug point for handling outbound requests. WebSphere Application Server uses this plug point to create the serialized information that is sent downstream based on the invocation Subject passed in and other security context information such as propagation tokens. A custom login module can use this plug point to change the identity. For more information, see “Configuring outbound identity mapping to a different target realm” on page 1540. Figure 2 shows where you can plug in custom login modules and shows where the wsMapCSIv2OutboundLoginModule login module is required.

  Figure 2

  For more information on the RMI_OUTBOUND login configuration, including its associated callbacks, see "RMI_OUTBOUND" in System login configuration entry settings for Java Authentication and Authorization Service.

- Handle inbound authentication for enterprise bean requests with the RMI_INBOUND login configuration.

  The RMI_INBOUND login configuration is a plug point that handles inbound authentication for enterprise bean requests. WebSphere Application Server uses this plug point for either an initial login or a propagation login. For more information about these two login types, see “Security attribute propagation” on page 1544. During a propagation login, this plug point is used to deserialize the information that is received from an upstream server. A custom login module can use this plug point to change the identity, handle custom tokens, add custom objects into the Subject, and so on. For more information on

changing the identity using a Hashtable object, which is referenced in figure 3, see “Configuring inbound identity mapping” on page 1533. Figure 3 shows where you can plug in custom login modules and shows that the ltpaLoginModule and the wsMapDefaultInboundLoginModule login modules are required.

Figure 3

For more information on the RMI_INBOUND login configuration, including its associated callbacks, see "RMI_INBOUND, WEB_INBOUND, DEFAULT" in System login configuration entry settings for Java Authentication and Authorization Service.

- Handle all other types of authentication requests with the DEFAULT login configuration. **DEFAULT login configuration**

  The DEFAULT login configuration is a plug point that handles all of the other types of authentication requests, including administrative SOAP requests and internal authentication of the server ID. Propagation logins typically do not occur at this plug point.

  For more information on the DEFAULT login configuration including its associated callbacks, see "RMI_INBOUND, WEB_INBOUND, DEFAULT" in System login configuration entry settings for Java Authentication and Authorization Service.

- Develop login configuration logic to know when specific information is present and how to use the information. **Writing a login module**

  When you write a login module that plugs into a WebSphere Application Server application login or system login configuration, read the JAAS programming model, which is located at: http://java.sun.com/products/jaas. The JAAS programming model provides basic information about JAAS. However, before writing a login module for the WebSphere Application Server environment, read the following sections in this article:

  – Useable callbacks
  – Shared state variables
  – Initial versus propagation logins
  – Sample custom login module

  Useable Callbacks

  Each login configuration must document the callbacks that are recognized by the login configuration. However, the callbacks are not always passed data. The login configuration must contain logic to know when specific information is present and how to use the information. For example, if you write a custom login module that can plug into all four of the pre-configured system login configurations mentioned previously, three sets of callbacks might be presented to authenticate a request. Other callbacks might be present for other reasons, including propagation and making other information available to the login configuration.

  Login information can be presented in the following combinations:

  User name (NameCallback) and password (PasswordCallback)
  This information is a typical authentication combination.

  User name only (NameCallback)
  This information is used for identity assertion, trust association interceptor (TAI) logins, and certificate logins.

  Token (WSCredTokenCallbackImpl)
  This information is for Lightweight Third Party Authentication (LTPA) token validation.

  Propagation token list (WSTokenHolderCallback)
  This information is used for a propagation login.

  The first three combinations are used for typical authentication. However, when the WSTokenHolderCallback callback is present in addition to one of the first three information combinations, the login is called a *propagation login*. A propagation login means that some security attributes are propagated to this server from another server. The servers can reuse these security attributes if the authentication information validates successfully. In some cases, a

WSTokenHolderCallback callback might not have sufficient attributes for a full login. Check the requiresLogin method on the WSTokenHolderCallback callback to determine if a new login is required. You can always ignore the information returned by the requiresLogin method, but, as a result, you might duplicate information. The following list contains the callbacks that might be present in the system login configurations. The list includes the callback name and a description of their responsibility.

callbacks[0] = new javax.security.auth.callback.NameCallback("Username: ");
This callback handler collects the user name for the login. The result can be the user name for a basic authentication login (user name and password) or a user name for an identity assertion login.

callbacks[1] = new javax.security.auth.callback.PasswordCallback("Password: ", false);
This callback handler collects the password for the login.

callbacks[2] = new com.ibm.websphere.security.auth.callback.WSCredTokenCallbackImpl("Credential Token: ");
This callback handler collects the Lightweight Third Party Authentication (LTPA) token or other token type for the login. This callback handler is typically present when a user name and password are not present.

callbacks[3] = new com.ibm.wsspi.security.auth.callback.WSTokenHolderCallback("Authz Token List: ");
This callback handler collects the ArrayList of TokenHolder objects that are returned from a call to the WSOpaqueTokenHelper.createTokenHolderListFromOpaqueToken API using the Common Secure Interoperability Version 2 (CSIv2) authorization token as input.

callbacks[4] = new com.ibm.websphere.security.auth.callback.WSServletRequestCallback("HttpServletRequest: ");
This callback handler collects the HTTP servlet request object, if present. This callback handler enables login modules to get information from the HTTP request for use in the login, and is presented from the WEB_INBOUND login configuration only.

callbacks[5] = new com.ibm.websphere.security.auth.callback.WSServletResponseCallback("HttpServletResponse: ");
This callback handler collects the HTTP servlet response object, if present. This callback handler enables login modules to put information into the HTTP response as a result of the login. An example of this situation might be adding the SingleSignonCookie cookie to the response.This callback handler is presented from the WEB_INBOUND login configuration only.

callbacks[6] = new com.ibm.websphere.security.auth.callback.WSAppContextCallback("ApplicationContextCallback: ");
This callback handler collects the web application context that is used during the login. This callback handler consists of a HashMap object, which contains the application name and the redirect web address, if present. The callback handler is presented from the WEB_INBOUND login configuration only.

callbacks[7] = new WSRealmNameCallbackImpl("Realm Name: ", *default_realm***);**
This callback handler collects the realm name for the login information. The realm information might not always be provided. If the realm information is not provided, assume that it is the current realm.

callbacks[8] = new WSX509CertificateChainCallback("X509Certificate[]: ");
This callback handler contains the certificate that was validated by Secure Sockets Layer (SSL) if the login source is an X509Certificate from SSL client authentication. The ltpaLoginModule calls the same mapping functions as WebSphere Application Server releases prior to version 6.1. However, having it passed into the login gives a custom login module the opportunity to map the certificate in a custom way. Then, it performs a Hashtable login. See “Configuring inbound identity mapping” on page 1533 for more information on a Hashtable login.

- Use shared state variables to share information between login modules during the login phase.

If you want to access the objects that WebSphere Application Server creates during a login, refer to the following shared state variables. The variables are set in the following login modules: ltpaLoginModule, swamLoginModule, and wsMapDefaultInboundLoginModule.

Shared state variable
com.ibm.wsspi.security.auth.callback.Constants.WSPRINCIPAL_KEY

Purpose
Specifies the com.ibm.websphere.security.auth.WSPrincipal object. See the WebSphere Application Server API documentation for application programming interface (API) usage. This shared state variable is for read-only purposes. Do not set this variable in the shared state for custom login modules.

The login module in which variables are set
ltpaLoginModule, swamLoginModule, and wsMapDefaultInboundLoginModule

Shared state variable
com.ibm.wsspi.security.auth.callback.Constants.WSCREDENTIAL_KEY

Purpose
Specifies the com.ibm.websphere.security.cred.WSCredential object. See the WebSphere Application Server API documentation for API usage. This shared state variable is for read-only purposes. Do not set this variable in the shared state for custom login modules.

Login module in which variables are set
wsMapDefaultInboundLoginModule

Shared state variable
com.ibm.wsspi.security.auth.callback.Constants.WSAUTHZTOKEN_KEY

Purpose
Specifies the default com.ibm.wsspi.security.token.AuthorizationToken object. Login modules can use this object to set custom attributes plugged in after the wsMapDefaultInboundLoginModule login module. The information set here is propagated downstream and is available to the application. See the WebSphere Application Server API documentation for API usage.

Initial versus propagation logins

As mentioned previously, some logins are considered initial logins because of the following reasons:

– It is the first time authentication information is presented to WebSphere Application Server.

– The login information is received from a server that does not propagate security attributes so this information must be gathered from a user registry.

Other logins are considered propagation logins when a WSTokenHolderCallback callback is present and contains sufficient information from a sending server to recreate all the required objects needed by WebSphere Application Server runtime. In cases where there is sufficient information for the WebSphere Application Server runtime, the information you might add to the Subject is likely to exist from the previous login. To verify if your object is present, you can get access to the ArrayList object that is present in the WSTokenHolderCallback callback, and search through this list looking at each TokenHolder getName method. This search is used to determine if WebSphere Application Server is deserializing your custom object during this login. Check the class name returned from the getName method using the String startsWith method because the runtime might add additional information at the end of the name to know which Subject is set to add the custom object after deserialization.

- Code your login() method to determine when sufficient information is present.

  The following code snippet can be used in your login() method to determine when sufficient information is present. For another example, see “Configuring inbound identity mapping” on page 1533.

```
// This is a hint provided by WebSphere Application Server that
// sufficient propagation information does not exist and, therefore,
// a login is required to provide the sufficient information. In this
// situation, a Hashtable login might be used.
boolean requiresLogin = ((com.ibm.wsspi.security.auth.callback.
WSTokenHolderCallback) callbacks[1]).requiresLogin();
```

```
if (requiresLogin)
{
// Check to see if your object exists in the TokenHolder list,
if not, add it.
java.util.ArrayList authzTokenList = ((WSTokenHolderCallback) callbacks[6]).
getTokenHolderList();boolean found = false;

if (authzTokenList != null)
{
Iterator tokenListIterator = authzTokenList.iterator();

while (tokenListIterator.hasNext())
{
com.ibm.wsspi.security.token.TokenHolder th = (com.ibm.wsspi.security.token.
TokenHolder) tokenListIterator.next();

if (th != null && th.getName().startsWith("com.acme.myCustomClass"))
{
found=true;
break;
}
}
if (!found)
{
// go ahead and add your custom object.
}
}
}
else
{
// This code indicates that sufficient propagation information is present.
// User registry calls are not needed by WebSphere Application Server to
// create a valid Subject. This code might be a no-op in your login module.
}
```

Sample custom login module

You can use the following sample to get ideas on how to use some of the callbacks and shared state variables.

```
{
 // Defines your login module variables
 com.ibm.wsspi.security.token.AuthenticationToken customAuthzToken = null;
 com.ibm.wsspi.security.token.AuthenticationToken defaultAuthzToken = null;
 com.ibm.websphere.security.cred.WSCredential credential = null;
 com.ibm.websphere.security.auth.WSPrincipal principal = null;
 private javax.security.auth.Subject _subject;
 private javax.security.auth.callback.CallbackHandler _callbackHandler;
 private java.util.Map _sharedState;
 private java.util.Map _options;

 public void initialize(Subject subject, CallbackHandler callbackHandler,
     Map sharedState, Map options)
 {
  _subject = subject;
  _callbackHandler = callbackHandler;
  _sharedState = sharedState;
  _options = options;
 }

 public boolean login() throws LoginException
 {
  boolean succeeded = true;

  // Gets the CALLBACK information
  javax.security.auth.callback.Callback callbacks[] = new javax.security.
        auth.callback.Callback[7];
  callbacks[0] = new javax.security.auth.callback.NameCallback(
        "Username: ");
  callbacks[1] = new javax.security.auth.callback.PasswordCallback(
        "Password: ", false);
  callbacks[2] = new com.ibm.websphere.security.auth.callback.
        WSCredTokenCallbackImpl ("Credential Token: ");
  callbacks[3] = new com.ibm.wsspi.security.auth.callback.
        WSServletRequestCallback ("HttpServletRequest: ");
  callbacks[4] = new com.ibm.wsspi.security.auth.callback.
        WSServletResponseCallback ("HttpServletResponse: ");
  callbacks[5] = new com.ibm.wsspi.security.auth.callback.
        WSAppContextCallback ("ApplicationContextCallback: ");
  callbacks[6] = new com.ibm.wsspi.security.auth.callback.
         WSTokenHolderCallback ("Authz Token List: ");

  try
  {
   callbackHandler.handle(callbacks);
  }
  catch (Exception e)
  {
   // Handles exceptions
```

```
 throw new WSLoginFailedException (e.getMessage(), e);
}

// Sees which callbacks contain information
uid = ((NameCallback) callbacks[0]).getName();
char password[] = ((PasswordCallback) callbacks[1]).getPassword();
byte[] credToken = ((WSCredTokenCallbackImpl) callbacks[2]).getCredToken();
javax.servlet.http.HttpServletRequest request = ((WSServletRequestCallback)
      callbacks[3]).getHttpServletRequest();
javax.servlet.http.HttpServletResponse response = ((WSServletResponseCallback)
      callbacks[4]).getHttpServletResponse();
java.util.Map appContext = ((WSAppContextCallback)
      callbacks[5]).getContext();
java.util.List authzTokenList = ((WSTokenHolderCallback)
      callbacks[6]).getTokenHolderList();

// Gets the SHARED STATE information
principal = (WSPrincipal) _sharedState.get(com.ibm.wsspi.security.
      auth.callback.Constants.WSPRINCIPAL_KEY);
credential = (WSCredential) _sharedState.get(com.ibm.wsspi.security.
      auth.callback.Constants.WSCREDENTIAL_KEY);
defaultAuthzToken = (AuthorizationToken) _sharedState.get(com.ibm.
      wsspi.security.auth.callback.Constants.WSAUTHZTOKEN_KEY);

   // What you tend to do with this information depends upon the scenario
   // that you are trying to accomplish. This example demonstrates how to
   // access various different information:
   // - Determine if a login is initial versus propagation
   // - Deserialize a custom authorization token (For more information, see
   //   "Security attribute propagation" on page 1544
   // - Add a new custom authorization token (For more information, see
   //   "Security attribute propagation" on page 1544
   // - Look for a WSCredential and read attributes, if found.
   // - Look for a WSPrincipal and read attributes, if found.
   // - Look for a default AuthorizationToken and add attributes, if found.
   // - Read the header attributes from the HttpServletRequest, if found.
   // - Add an attribute to the HttpServletResponse, if found.
   // - Get the web application name from the appContext, if found.

   // - Determines if a login is initial versus propagation. This is most
   //  useful when login module is first.
boolean requiresLogin = ((WSTokenHolderCallback) callbacks[6]).requiresLogin();

// initial login - asserts privilege attributes based on user identity
if (requiresLogin)
{

 // If you are validating a token from another server, there is an
     // application programming interface  (API) to get the uniqueID from it.
 if (credToken != null && uid == null)
 {
  try
  {
   String uniqueID = WSSecurityPropagationHelper.
              validateLTPAToken(credToken);
   String realm = WSSecurityPropagationHelper.getRealmFromUniqueID
               (uniqueID);
          // Now set it to the UID so you can use that to either map or
          // login with.
   uid = WSSecurityPropagationHelper.getUserFromUniqueID (uniqueID);
  }
  catch (Exception e)
  {
   // handle exception
  }
 }
     // Adds a Hashtable to shared state.
     // Note: You can perform custom mapping on the NameCallback value returned
     // to change the identity based upon your own mapping rules.
 uid = mapUser (uid);

 // Gets the default InitialContext for this server.
 javax.naming.InitialContext ctx = new javax.naming.InitialContext();

 // Gets the local UserRegistry object.
 com.ibm.websphere.security.UserRegistry reg = (com.ibm.websphere.security.
          UserRegistry) ctx.lookup("UserRegistry");

     // Gets the user registry uniqueID based on the uid specified in the
     // NameCallback.
 String uniqueid = reg.getUniqueUserId(uid);
  uid = WSSecurityPropagationHelper.getUserFromUniqueID (uniqueID);

 // Gets the display name from the user registry based on the uniqueID.
 String securityName = reg.getUserSecurityName(uid);

 // Gets the groups associated with this uniqueID.
 java.util.List groupList = reg.getUniqueGroupIds(uid);

     // Creates the java.util.Hashtable with the information you gathered from
```

```
    // the UserRegistry.
 java.util.Hashtable hashtable = new java.util.Hashtable();
 hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
         WSCREDENTIAL_UNIQUEID, uniqueid);
 hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
         WSCREDENTIAL_SECURITYNAME, securityName);
 hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
         WSCREDENTIAL_GROUPS, groupList);

     // Adds a cache key that is used as part of the lookup mechanism for
     // the created Subject. The cache key can be an Object, but should
     // implement the toString() method. Make sure the cacheKey contains
     // enough information to scope it to the user and any additional
     // attributes that you use. If you do not specify this property the
     // Subject is scoped to the WSCREDENTIAL_UNIQUEID returned, by default.
 hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
        WSCREDENTIAL_CACHE_KEY,
   "myCustomAttribute" + uniqueid);

  // Adds the hashtable to the sharedState of the Subject.
  _sharedState.put(com.ibm.wsspi.security.token.AttributeNameConstants.
           WSCREDENTIAL_PROPERTIES_KEY,hashtable);
}
// propagation login - process propagated tokens
else
{
 // - Deserializes a custom authorization token. For more information, see
     //    "Security attribute propagation" on page 1544.
     //    This can be done at any login module plug in point (first,
     //     middle, or last).
 if (authzTokenList != null)
 {
  // Iterates through the list looking for your custom token
  for (int i=0; i<authzTokenList.size(); i++)
  {
   TokenHolder tokenHolder = (TokenHolder)authzTokenList.get(i);

   // Looks for the name and version of your custom AuthorizationToken
          // implementation
   if (tokenHolder.getName().equals("com.ibm.websphere.security.token.
             CustomAuthorizationTokenImpl") && tokenHolder.getVersion() == 1)
   {
    // Passes the bytes into your custom AuthorizationToken constructor
            // to deserialize
    customAuthzToken = new
     com.ibm.websphere.security.token.
                 CustomAuthorizationTokenImpl(tokenHolder.getBytes());

   }
  }
 }
      // - Adds a new custom authorization token (For more information,
      //   see "Security attribute propagation" on page 1544)
      //    This can be done at any login module plug in point (first, middle,
      //    or last).
 else
 {
  // Gets the PRINCIPAL from the default AuthenticationToken. This must
        // match all of the tokens.
  defaultAuthToken = (com.ibm.wsspi.security.token.AuthenticationToken)
   sharedState.get(com.ibm.wsspi.security.auth.callback.Constants.
            WSAUTHTOKEN_KEY);
  String principal = defaultAuthToken.getPrincipal();

  // Adds a new custom authorization token. This is an initial login.
        // Pass the principal into the constructor
  customAuthzToken = new com.ibm.websphere.security.token.
            CustomAuthorizationTokenImpl(principal);

  // Adds any initial attributes
  if (customAuthzToken != null)
  {
   customAuthzToken.addAttribute("key1", "value1");
   customAuthzToken.addAttribute("key1", "value2");
   customAuthzToken.addAttribute("key2", "value1");
   customAuthzToken.addAttribute("key3", "something different");
  }
 }
}

// - Looks for a WSCredential and read attributes, if found.
//   This is most useful when plugged in as the last login module.
if (credential != null)
{
 try
 {
  // Reads some data from the credential
  String securityName = credential.getSecurityName();
  java.util.ArrayList = credential.getGroupIds();
 }
```

```
  catch (Exception e)
  {
   // Handles exceptions
   throw new WSLoginFailedException (e.getMessage(), e);
  }
 }

 // - Looks for a WSPrincipal and read attributes, if found.
 //   This is most useful when plugged as the last login module.
 if (principal != null)
 {
  try
  {
   // Reads some data from the principal
   String principalName = principal.getName();
  }
  catch (Exception e)
  {
   // Handles exceptions
   throw new WSLoginFailedException (e.getMessage(), e);
  }
 }

 // - Looks for a default AuthorizationToken and add attributes, if found.
 //   This is most useful when plugged in as the last login module.
 if (defaultAuthzToken != null)
 {
  try
  {
   // Reads some data from the defaultAuthzToken
   String[] myCustomValue = defaultAuthzToken.getAttributes ("myKey");
   // Adds some data if not present in the defaultAuthzToken
   if (myCustomValue == null)
    defaultAuthzToken.addAttribute ("myKey", "myCustomData");
  }
  catch (Exception e)
  {
   // Handles exceptions
   throw new WSLoginFailedException (e.getMessage(), e);
  }
 }

 // - Reads the header attributes from the HttpServletRequest, if found.
 //  This can be done at any login module plug in point (first, middle,
    //  or last).
 if (request != null)
 {
  java.util.Enumeration headerEnum = request.getHeaders();
  while (headerEnum.hasMoreElements())
  {
   System.out.println ("Header element: " + (String)headerEnum.nextElement());
  }
 }

 // - Adds an attribute to the HttpServletResponse, if found
    //  This can be done at any login module plug in point (first, middle,
    //  or last).
 if (response != null)
 {
  response.addHeader ("myKey", "myValue");
 }

 // - Gets the web application name from the appContext, if found
 //   This can be done at any login module plug in point (first, middle,
    //   or last).
 if (appContext != null)
 {
  String appName = (String) appContext.get(com.ibm.wsspi.security.auth.
         callback.Constants.WEB_APP_NAME);
 }

 return succeeded;
}

public boolean commit() throws LoginException
{
 boolean succeeded = true;

 // Add any objects here that you have created and belong in the
 // Subject. Make sure the objects are not already added. If you added
 // any sharedState variables, remove them before you exit. If the abort()
    // method gets called, make sure you cleanup anything added to the
 // Subject here.

 if (customAuthzToken != null)
 {
  // Sets the customAuthzToken token into the Subject
  try
  {
   // Do this in a doPrivileged code block so that application code
```

```
        // does not need to add additional permissions
   java.security.AccessController.doPrivileged(new java.security.PrivilegedAction()
   {
    public Object run()
    {
     try
     {
      // Adds the custom authorization token if it is not
                //  null and not already in the Subject
                              if ((customAuthzTokenPriv != null) &&
        (!_subject.getPrivateCredentials().contains(customAuthzTokenPriv)))
      {
       _subject.getPrivateCredentials().add(customAuthzTokenPriv);
      }
     }
     catch (Exception e)
     {
      throw new WSLoginFailedException (e.getMessage(), e);
     }

     return null;
    }
   });
  }
  catch (Exception e)
  {
   throw new WSLoginFailedException (e.getMessage(), e);
  }
 }

 return succeeded;
}

public boolean abort() throws LoginException
{
 boolean succeeded = true;

 // Makes sure to remove all objects that have already been added (both into the
    // Subject and shared state).

 if (customAuthzToken != null)
 {
  // remove the customAuthzToken token from the Subject
  try
  {
   final AuthorizationToken customAuthzTokenPriv = customAuthzToken;
   // Do this in a doPrivileged block so that application code does not need
         // to add additional permissions
   java.security.AccessController.doPrivileged(new java.security.PrivilegedAction()
   {
    public Object run()
    {
     try
     {
      // Removes the custom authorization token if it is not
                // null and not already in the Subject
                    if ((customAuthzTokenPriv != null) &&
                     (_subject.getPrivateCredentials().
                      contains(customAuthzTokenPriv)))
      {
       _subject.getPrivateCredentials().
                   remove(customAuthzTokenPriv);
      }
     }
     catch (Exception e)
     {
      throw new WSLoginFailedException (e.getMessage(), e);
     }

     return null;
    }
   });
  }
  catch (Exception e)
  {
   throw new WSLoginFailedException (e.getMessage(), e);
  }
 }

 return succeeded;
}


public boolean logout() throws LoginException
{
 boolean succeeded = true;

 // Makes sure to remove all objects that have already been added
    // (both into the Subject and shared state).
```

```
  if (customAuthzToken != null)
  {
   // Removes the customAuthzToken token from the Subject
   try
   {
    final AuthorizationToken customAuthzTokenPriv = customAuthzToken;
    // Do this in a doPrivileged code block so that application code does
          // not need to add additional permissions
    java.security.AccessController.doPrivileged(new java.security.
            PrivilegedAction()
    {
     public Object run()
     {
      try
      {
       // Removes the custom authorization token if it is not null and not
                 //  already in the Subject
                     if ((customAuthzTokenPriv != null) && (_subject.
                     getPrivateCredentials().
                     contains(customAuthzTokenPriv)))
       {
        _subject.getPrivateCredentials().remove(customAuthzTokenPriv);
       }
      }
      catch (Exception e)
      {
       throw new WSLoginFailedException (e.getMessage(), e);
      }

      return null;
     }
    });
   }
   catch (Exception e)
   {
    throw new WSLoginFailedException (e.getMessage(), e);
   }
  }

  return succeeded;

 }

}
```

- Configure the system login for your custom login module.

  After developing your custom login module for a system login configuration, you can configure the system login using either the administrative console or using the wsadmin utility. To configure the system login using the administrative console, click **Security > Global security**. Under Java Authentication and Authorization Service, click **System logins**. For more information on using the wsadmin utility for system login configuration, see Customizing a server-side Java Authentication and Authorization Service authentication and login configuration. Also refer to that article for information on system login modules and to determine whether to add additional login modules.

Customizing application login with Java Authentication and Authorization Service:

Using Java Authentication and Authorization Service (JAAS), you can customize your application login.

About this task

Java Authentication and Authorization Service (JAAS) is an API that enables applications to access authentication and access control services without being tied to those services. The following topics explaining customizing your application with JAAS are covered in this section:

Procedure

1. Develop programmatic logins with JAAS.

   You can develop programmatic logins with JAAS, which represents the strategic application programming interfaces (API) for authentication.
2. Configure programmatic logins with JAAS.

   A new JAAS login configuration can be added and modified using the administrative console. The changes are saved in the cell-level security document and are available to all managed application servers.

3. Customize an application login to perform an identity assertion using JAAS.

   Using the JAAS login framework, you can create a JAAS login configuration that can be used to perform login to an identity assertion.
4. Configure a server-side JAAS authentication and login configuration.

   WebSphere Application Server supports plugging in a custom JAAS login module before or after the WebSphere Application Server system login module. However, WebSphere Application Server does not support the replacement of the WebSphere Application Server system login modules, which are used to create the WSCredential credential and WSPrincipal principal in the Subject. By using a custom login module, you can either make additional authentication decisions or add information to the Subject to make additional, potentially finer-grained, authorization decisions inside a Java Platform, Enterprise Edition (Java EE) application.

Enabling identity assertion with trust validation using JAAS:

By enabling identity assertion with trust validation, an application can use the JAAS login configuration to perform a programmatic identity assertion.

About this task

To enable an identity assertion with trust validation, follow these steps:

Procedure

1. Create a custom login module to perform a trust validation. The login module must set trust and identity information in the shared state, which is then passed on to the IdentityAssertionLoginModule. The trust and identity information is stored in a map in the shared state under the key, com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state. If this key is missing from the shared state, a WSLoginFailedException error is thrown by the IdentityAssertionLoginModule module. The custom login module should include the following:
   - A trust key named com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trust. If the trust key is set to `true`, trust is established. If the trust key is set to `false`, the IdentityAssertionLoginModule module creates a WSLoginFailedException error.
   - The identity of the java.security.Principal type set in the com.ibm.wsspi.security.common.auth.module.IdenityAssertionLoginModule.principal key.
   - The identity in the form of a java,security.cert.X509Certificate[] certificate set in the com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates key.

   Note: If both a principal and a certificate are supplied, the principal is used, and a warning is issued.
2. Create a new Java Authentication and Authorization Service (JAAS) configuration for application logins. It contains the user-implemented trust validation custom login module and the IdentityAssertionLoginModule module. To configure an application login configuration from the administrative console, complete the following steps:
   a. Click **Security > Global security**.
   b. Under Java Authentication and Authorization Service, click **Application logins > New**.
   c. Supply the JAAS configuration with an alias, and then click **Apply**.
   d. Under Additional properties, click **JAAS Login Modules > New**.
   e. Enter the module class name of the user-implemented trust validation custom login module, and then click **Apply**.
   f. Enter the com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule module class name.
   g. Make sure that the module class name classes are in the correct order. The user-implemented trust validation login module must be the first class in the list, and the IdentityAssertionLoginModule module must be the second class.

h. Click **Save**. The new JAAS configuration is used by the application to perform an identity assertion.

What to do next

An application can now use the JAAS login configuration to perform a programmatic identity assertion. The application can create a login context for the JAAS configuration created in step 2, then login to that login context with the identity it asserts to. If the login is successful, that identity can be set in the current running process, as in the following example:

```
MyCallbackHandler handler = new MyCallbackHandler(new MyPrincipal("Joe"));
LoginContext lc = new LoginContext("MyAppLoginConfig", handler);
lc.login();  //assume successful
Subject s = lc.getSubject();
WSSubject.setRunAsSubject(s);
// From here on, the runas identity is "Joe"
```

Performing identity mapping for authorization across servers in different realms

Identity mapping is a one-to-one mapping of a user identity between two servers so that the proper authorization decisions are made by downstream servers. Identity mapping is necessary when the integration of servers is needed, but the user registries are different and not shared between the systems.

About this task

In most cases, requests flow downstream between two servers that are part of the same security domain. In WebSphere Application Server, two servers that are members of the same cell are also members of the same security domain. In the same cell, the two servers have the same user registry and the same Lightweight Third Party Authentication (LTPA) keys for token encryption. These two commonalities ensure that the LTPA token, among other user attributes, which flows between the two servers, not only can be decrypted and validated, but also the user identity in the token can be mapped to attributes that are recognized by the authorization engine.

The most reliable and recommended configuration involves two servers within the same cell. However, sometimes you need to integrate multiple systems that cannot use the same user registry. When the user registries are different between two servers, the security domain or realm of the target server does not match the security domain of the sending server.

WebSphere Application Server enables mapping to occur either before sending the request outbound or before enabling the existing security credentials to flow to the target server. The credentials are mapped inbound with the specification that the target realm is trusted.

An alternative to mapping is to send the user identity without the token or the password to a target server without actually mapping the identity. The use of the user identity is based on trust between the two servers. Use Common Secure Interoperability Version 2 (CSIv2) identity assertion. When enabled, the server sends just the X.509 certificate, principal name, or distinguished name (DN) based upon what was used by the original client to perform the initial authentication. During CSIv2 identity assertion, trust is established between WebSphere Application Servers.

The user identity must exist in the target user registry for identity assertion to work. This process can also enable interoperability between other Java 2 Platform, Enterprise Edition (J2EE) Version 1.4 and higher compliant application servers. If both the sending server and target servers have identity assertion configured, WebSphere Application Server always uses this method of authentication, even when both servers are in the same security domain. For more information on CSIv2 identity assertion, see “Identity assertion to the downstream server” on page 1597.

When the user identity is not present in the user registry of the target server, identity mapping must occur either before the request is sent outbound or when the request comes inbound. This decision depends upon your environment and requirements. However, it is typically easier to map the user identity before the request is sent outbound for the following reasons:

- You know the user identity of the existing credential as it comes from the user registry of the sending server.
- You do not have to worry about sharing Lightweight Third Party Authentication (LTPA) keys with the other target realm because you are not mapping the identity to LTPA credentials. Typically, you are mapping the identity to a user ID and password that are present in the user registry of the target realm.

When you do perform outbound mapping, in most cases, it is recommended that you use Secure Sockets Layer (SSL) to protect the integrity and confidentiality of the security information sent across the network. If LTPA keys are not shared between servers, an LTPA token cannot be validated at the inbound server. In this case, outbound mapping is necessary because the user identity cannot be determined at the inbound server to do inbound mapping. For more information, see “Configuring outbound identity mapping to a different target realm” on page 1540.

When you need inbound mapping, potentially due to the mapping capabilities of the inbound server, you must ensure that both servers have the same LTPA keys so that you can get access to the user identity. Typically, in secure communications between servers, an LTPA token is passed into the WSCredTokenCallback callback of the inbound JAAS login configuration for the purposes of client authentication. A method is available that enables you to open the LTPA token, if valid, and get access to the user unique ID so that mapping can be performed. For more information, see “Configuring inbound identity mapping.” In other cases, such as identity assertion, you might receive a user name in the NameCallback callback of the inbound login configuration that enables you to map the identity.

The following topics are covered in this section:

Procedure

- Configuring inbound identity mapping For inbound identity mapping, you can write a custom login module and configure WebSphere Application Server to run the login module first within the system login configurations. Consider the following steps when you write your custom login module: “Configuring inbound identity mapping.”
- Configuring outbound identity mapping to a different target realm By default, when WebSphere Application Server makes an outbound request from one server to another server in a different security realm, the request is rejected. This topic details alternatives for enabling one server to send outbound requests to a target server in a different realm. For more information, see “Configuring outbound identity mapping to a different target realm” on page 1540

Configuring inbound identity mapping

For inbound identity mapping, write a custom login module and configure WebSphere Application Server to run the login module first within the system login configurations. Consider the following steps when you write your custom login module.

Procedure

1. Get the inbound user identity from the callbacks and map the identity, if necessary This step occurs in the login method of the login module. A valid authentication has either or both NameCallback and the WSCredTokenCallback callbacks present. The following code sample shows you how to determine the user identity:

```
javax.security.auth.callback.Callback callbacks[] =
   new javax.security.auth.callback.Callback[3];
 callbacks[0] = new javax.security.auth.callback.NameCallback("");
 callbacks[1] = new javax.security.auth.callback.PasswordCallback
     ("Password: ", false);
 callbacks[2] = new com.ibm.websphere.security.auth.callback.
     WSCredTokenCallbackImpl("");
```

```
callbacks[3] = new com.ibm.wsspi.security.auth.callback.
    WSTokenHolderCallback("");

try
{
 callbackHandler.handle(callbacks);
}
catch (Exception e)
{
 // Handles exceptions
 throw new WSLoginFailedException (e.getMessage(), e);
}

// Shows which callbacks contain information
boolean identitySwitched = false;
String uid = ((NameCallback) callbacks[0]).getName();
char password[] = ((PasswordCallback) callbacks[1]).getPassword();
byte[] credToken = ((WSCredTokenCallbackImpl) callbacks[2]).getCredToken();
java.util.List authzTokenList = ((WSTokenHolderCallback)
    callbacks[3]).getTokenHolderList();

if (credToken != null)
{
 try
 {
  String uniqueID = WSSecurityPropagationHelper.validateLTPAToken(credToken);
  String realm = WSSecurityPropagationHelper.getRealmFromUniqueID (uniqueID);
      // Now set the string to the UID so that you can use the result for either
      // mapping or logging in.
  uid = WSSecurityPropagationHelper.getUserFromUniqueID (uniqueID);
 }
 catch (Exception e)
 {
  // Handles the exception
 }
}
else if (uid == null)
{
    // Throws an exception if authentication data is not valid.
    // You must have either UID or CredToken
 throw new WSLoginFailedException("invalid authentication data.");
}
else if (uid != null && password != null)
{
    // This is a typical authentication. You can choose to map this ID to
    // another ID or you can skip it and allow WebSphere Application Server
    // to log in for you. When passwords are presented, be very careful to not
    // validate the password because this is the initial authentication.

 return true;
}

   // If desired, map this uid to something else and set the identitySwitched
   // boolean. If the identity was changed, clear the propagated attributes
   // below so they are not used incorrectly.
uid = myCustomMappingRoutine (uid);

   // Clear the propagated attributes because they are no longer applicable
   // to the new identity
if (identitySwitched)
{
 ((WSTokenHolderCallback) callbacks[3]).setTokenHolderList(null);
}
```

2. Check to see if attribute propagation occurred and if the attributes for the user are already present when the identity remains the same. Check to see if the user attributes are already present from the sending server to avoid duplicate calls to the user registry lookup. To check for the user attributes, use

a method on the WSTokenHolderCallback callback that analyzes the information present in the callback to determine if the information is sufficient for WebSphere Application Server to create a Subject. The following code sample checks for the user attributes:

```
boolean requiresLogin =
((com.ibm.wsspi.security.auth.callback.WSTokenHolderCallback)
callbacks[2]).getrequiresLogin();
```

If sufficient attributes are not present to form the WSCredential and the WSPrincipal objects that are needed to perform authorization, the previous code sample returns a `true` result. When the result is `false`, you can choose to discontinue processing as the necessary information exists to create the Subject without performing additional remote user registry calls.

3. Optional: Look up the required attributes from the user registry, put the attributes in a hashtable, and add the hashtable to the shared state. If the identity is switched in this login module, you must complete the following steps:
   a. Create the hashtable of attributes, as shown in the following example.
   b. Add the hashtable to the shared state.

   If the identity is not switched, but the value of the requiresLogin code sample shown previously is `true`, you can create the hashtable of attributes. However, you are not required to create a hashtable in this situation as WebSphere Application Server handles the login for you. However, you might consider creating a hashtable to gather attributes in special cases where you are using your own special user registry. Creating a UserRegistry implementation, using a hashtable, and letting WebSphere Application Server gather the user attributes for you might be the easiest solution. The following table shows how to create a hashtable of user attributes:

```
if (requiresLogin || identitySwitched)
 {
  // Retrieves the default InitialContext for this server.
  javax.naming.InitialContext ctx = new javax.naming.InitialContext();

  // Retrieves the local UserRegistry implementation.
  com.ibm.websphere.security.UserRegistry reg = (com.ibm.websphere.
        security.UserRegistry)
  ctx.lookup("UserRegistry");

     // Retrieves the user registry uniqueID based on the uid specified
     // in the NameCallback.
  String uniqueid = reg.getUniqueUserId(uid);
   uid = WSSecurityPropagationHelper.getUserFromUniqueID (uniqueid);

     // Retrieves the display name from the user registry based on the uniqueID.
  String securityName = reg.getUserSecurityName(uid);

     // Retrieves the groups associated with the uniqueID.
  java.util.List groupList = reg.getUniqueGroupIds(uid);

     // Creates the java.util.Hashtable with the information that you gathered
     // from the UserRegistry implementation.
  java.util.Hashtable hashtable = new java.util.Hashtable();
  hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_UNIQUEID, uniqueid);
     hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_SECURITYNAME, securityName);
  hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_GROUPS, groupList);

     // Adds a cache key that is used as part of the lookup mechanism for
     // the created Subject. The cache key can be an object, but should have
     // an implemented toString method. Make sure that the cacheKey contains
     // enough information to scope it to the user and any additional attributes
     // that you are using. If you do not specify this property the Subject is
     // scoped to the returned WSCREDENTIAL_UNIQUEID, by default.
  hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
```

```
        WSCREDENTIAL_CACHE_KEY, "myCustomAttribute" + uniqueid);
  // Adds the hashtable to the sharedState of the Subject.
  _sharedState.put(com.ibm.wsspi.security.token.AttributeNameConstants.
        WSCREDENTIAL_PROPERTIES_KEY, hashtable);
 }
```

The following rules define in more detail how a hashtable login is performed. You must use a java.util.Hashtable object in either the Subject (public or private credential set) or the shared-state HashMap. The com.ibm.wsspi.security.token.AttributeNameConstants class defines the keys that contain the user information. If the Hashtable object is put into the shared state of the login context using a custom login module that is listed prior to the Lightweight Third Party Authentication (LTPA) login module, the value of the java.util.Hashtable object is searched using the following key within the shared-state hashMap:

Property
com.ibm.wsspi.security.cred.propertiesObject

Reference to the property
AttributeNameConstants.WSCREDENTIAL_PROPERTIES_KEY

Explanation
This key searches for the Hashtable object that contains the required properties in the shared state of the login context.

Expected result
A java.util.Hashtable object.

If a java.util.Hashtable object is found either inside the Subject or within the shared state area, verify that the following properties are present in the hashtable:

Property
com.ibm.wsspi.security.cred.uniqueId

Reference to the property
AttributeNameConstants.WSCREDENTIAL_UNIQUEID

Returns
java.util.String

Explanation
The value of the property must be a unique representation of the user. For the WebSphere Application Server default implementation, this property represents the information that is stored in the application authorization table. The information is located in the application deployment descriptor after it is deployed and user-to-role mapping is performed. See the expected format examples if the user to role mapping is performed using a lookup to a WebSphere Application Server user registry implementation.

If a third-party authorization provider overrides the user-to-role mapping, then the third-party authorization provider defines the format. To ensure compatibility with the WebSphere Application Server default implementation for the unique ID value, call the WebSphere Application Server public String getUniqueUserId(String userSecurityName) UserRegistry method.

Expected format examples

Table 107. Format examples.

This table gives some format examples when configuring inbound identity mapping.

Realm	Format (uniqueUserId)
Lightweight Directory Access Protocol (LDAP)	`ldaphost.austin.ibm.com:389/cn=user,o=ibm,c=us`
Windows	`MYWINHOST/S-1-5-21-963918322-163748893-4247568029-500`
UNIX	`MYUNIXHOST/32`

The com.ibm.wsspi.security.cred.uniqueId property is required.

Property
com.ibm.wsspi.security.cred.securityName

Reference to the property
AttributeNameConstants. WSCREDENTIAL_ SECURITYNAME

Returns
java.util.String

Explanation
This property searches for the securityName of the authentication user. This name is commonly called the *display name* or *short name*. WebSphere Application Server uses the securityName attribute for the getRemoteUser, getUserPrincipal and getCallerPrincipal application programming interfaces (APIs). To ensure compatibility with the WebSphere Application Server default implementation for the securityName value, call the WebSphere Application Server public String getUserSecurityName(String uniqueUserId) UserRegistry method.

Expected format examples

Table 108. Format examples. This table gives expected format examples.

Realm	Format (uniqueUserId)
LDAP	user (*LDAP UID*)
Windows	user (*Windows username*)
UNIX	user (*UNIX username*)

The com.ibm.wsspi.security.cred.securityName property is required.

Property
com.ibm.wsspi.security.cred.groups

Reference to the property
AttributeNameConstants. WSCREDENTIAL_GROUPS

Returns
java.util.ArrayList

Explanation
This key searches for the array list of groups to which the user belongs. The groups are specified in the *realm_name*/*user_name* format. The format of these groups is important as the groups are used by the WebSphere Application Server authorization engine for group-to-role mappings in the deployment descriptor. The format that is provided must match the format expected by the WebSphere Application Server default implementation. When you use a third-party authorization provider, you must use the format that is expected by the third-party provider. To ensure compatibility with the WebSphere Application Server default implementation for the unique group IDs value, call the WebSphere Application Server public List getUniqueGroupIds(String uniqueUserId) UserRegistry method.

Expected format examples for each group in the array list

Table 109. Format examples. This table gives expected format examples for each group in the array list.

Realm	Format
LDAP	`ldap1.austin.ibm.com:389/cn=group1,o=ibm,c=us`
Windows	`MYWINREALM/S-1-5-32-544`
UNIX	`MY/S-1-5-32-544`

The com.ibm.wsspi.security.cred.groups property is not required. A user is not required to have associated groups.

Property
com.ibm.wsspi.security.cred.cacheKey

Reference to the property
AttributeNameConstants. WSCREDENTIAL_CACHE_KEY

Returns
java.lang.Object

Explanation
This key property can specify an object that represents the unique properties of the login, including the user-specific information and the user dynamic attributes that might affect uniqueness. For example, when the user logs in from location A, which might affect their access control, the cache key needs to include location A so that the Subject that is received is the correct Subject for the current location.

This com.ibm.wsspi.security.cred.cacheKey property is not required. When this property is not specified, the cache lookup is the value that is specified for WSCREDENTIAL_UNIQUEID. When this information is found in the java.util.Hashtable object, WebSphere Application Server creates a Subject similar to the Subject that goes through the normal login process at least for LTPA. The new Subject contains a WSCredential object and a WSPrincipal object that is fully populated with the information found in the Hashtable object.

4. Add your custom login module into the RMI_INBOUND, WEB_INBOUND, and DEFAULT Java Authentication and Authorization Service (JAAS) system login configurations. Configure the RMI_INBOUND login configuration so that WebSphere Application Server loads your new custom login module first.
   a. Click **Security > Global security > Java Authentication and Authorization Service > System logins > RMI_INBOUND**
   b. Under Additional Properties, click **JAAS login modules > New** to add your login module to the RMI_INBOUND configuration.
   c. Return to the JAAS login modules panel for RMI_INBOUND.
   d. Click **Set order** to change the order that the login modules are loaded so that WebSphere Application Server loads your custom login module first. Use the **Move Up** or **Move Down** buttons to arrange the order of the login modules.
   e. Repeat the previous three steps for the WEB_INBOUND and DEFAULT login configurations.

Results

This process configures identity mapping for an inbound request.

Example

The “Example: Custom login module for inbound mapping” topic shows a custom login module that creates a java.util.Hashtable hashtable that is based on the specified NameCallback callback. The java.util.Hashtable hashtable is added to the sharedState java.util.Map map so that the WebSphere Application Server login modules can locate the information in the hashtable.

Example: Custom login module for inbound mapping:

This sample shows a custom login module that creates a java.util.Hashtable hashtable that is based on the specified NameCallback callback. The java.util.Hashtable hashtable is added to the sharedState java.util.Map map so that the WebSphere Application Server login modules can locate the information in the Hashtable.

```
public customLoginModule()
{

public void initialize(Subject subject, CallbackHandler callbackHandler,
   Map sharedState, Map options)
{
```

```
 // (For more information on initialization, see
 //  "Developing custom login modules for a system login configuration for JAAS" on page 1521.)
 _sharedState = sharedState;
}

public boolean login() throws LoginException
{
 // (For more information on what to do during login, see
 //  "Developing custom login modules for a system login configuration for JAAS" on page 1521.)

   // Handles the WSTokenHolderCallback to see if this is an initial or
   // propagation login.
 javax.security.auth.callback.Callback callbacks[] =
     new javax.security.auth.callback.Callback[3];
 callbacks[0] = new javax.security.auth.callback.NameCallback("");
 callbacks[1] = new javax.security.auth.callback.PasswordCallback(
     "Password: ", false);
 callbacks[2] = new com.ibm.websphere.security.auth.callback.
     WSCredTokenCallbackImpl("");
 callbacks[3] = new com.ibm.wsspi.security.auth.callback.
     WSTokenHolderCallback("");

 try
 {
  callbackHandler.handle(callbacks);
 }
 catch (Exception e)
 {
  // Handles the exception
 }

 // Determines which callbacks contain information
 boolean identitySwitched = false;
 String uid = ((NameCallback) callbacks[0]).getName();
 char password[] = ((PasswordCallback) callbacks[1]).getPassword();
 byte[] credToken = ((WSCredTokenCallbackImpl) callbacks[2]).getCredToken();
 java.util.List authzTokenList = ((WSTokenHolderCallback) callbacks[3]).
     getTokenHolderList();

 if (credToken != null)
 {
  try
  {
   String uniqueID = WSSecurityPropagationHelper.validateLTPAToken(credToken);
   String realm = WSSecurityPropagationHelper.getRealmFromUniqueID (uniqueID);
       // Set the string to the UID so you can use the information to either
       // map or login.
   uid = WSSecurityPropagationHelper.getUserFromUniqueID (uniqueid);
  }
  catch (Exception e)
  {
   // handle exception
  }
 }
 else if (uid == null)
 {
  // The authentication data is not valid. You must have either UID
    // or CredToken
  throw new WSLoginFailedException("invalid authentication data.");
 }
 else if (uid != null && password != null)
 {
     // This is a typical authentication. You can choose to map this ID to
     // another ID or you can skip it and allow WebSphere Application Server
     // to log in for you. When passwords are presented, be very careful not
     // to validate the password because this is the initial authentication.

  return true;
 }

  // You can map this uid to something else and set the identitySwitched
  // boolean. If the identity is changed, clear the following propagated
  // attributes so they are not used incorrectly.
 uid = myCustomMappingRoutine (uid);

 // Clear the propagated attributes because they no longer apply to the new identity
 if (identitySwitched)
 {
  ((WSTokenHolderCallback) callbacks[3]).setTokenHolderList(null);
 }
 boolean requiresLogin = ((com.ibm.wsspi.security.auth.callback.
     WSTokenHolderCallback) callbacks[2]).getRequiresLogin();

 if (requiresLogin || identitySwitched)
 {
  // Retrieves the default InitialContext for this server.
  javax.naming.InitialContext ctx = new javax.naming.InitialContext();

  // Retrieves the local UserRegistry object.
  com.ibm.websphere.security.UserRegistry reg =
```

```
        (com.ibm.websphere.security.UserRegistry) ctx.lookup("UserRegistry");

  // Retrieves the registry uniqueID based on the uid that is specified
     // in the NameCallback.
  String uniqueid = reg.getUniqueUserId(uid);
   uid = WSSecurityPropagationHelper.getUserFromUniqueID (uniqueid);

  // Retrieves the display name from the user registry based on the uniqueID.
  String securityName = reg.getUserSecurityName(uid);

  // Retrieves the groups associated with this uniqueID.
  java.util.List groupList = reg.getUniqueGroupIds(uid);

  // Creates the java.util.Hashtable with the information that you gathered
     // from the UserRegistry.
  java.util.Hashtable hashtable = new java.util.Hashtable();
  hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_UNIQUEID, uniqueid);
     hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_SECURITYNAME, securityName);
  hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_GROUPS, groupList);

  // Adds a cache key that is used as part of the lookup mechanism for
  // the created Subject. The cache key can be an object, but has
  // an implemented toString method. Make sure the cacheKey contains enough
  // information to scope it to the user and any additional attributes you are
     // using. If you do not specify this property, the Subject is scoped to the
  // WSCREDENTIAL_UNIQUEID returned, by default.
  hashtable.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_CACHE_KEY, "myCustomAttribute" + uniqueid);
  // Adds the hashtable to the shared state of the Subject.
  _sharedState.put(com.ibm.wsspi.security.token.AttributeNameConstants.
       WSCREDENTIAL_PROPERTIES_KEY, hashtable);
 }
 else if (requiresLogin == false)
 {
  // For more information on this section, see
     // "Security attribute propagation" on page 1544.
  // If you added a custom Token implementation, you can search through the
     // token holder list for it to deserialize.
  // Note: Any Java objects are automatically deserialized by
     // wsMapDefaultInboundLoginModule

  for (int i=0; i<authzTokenList.size(); i++)
  {
   TokenHolder tokenHolder = (TokenHolder) authzTokenList.get(i);
   if (tokenHolder.getName().equals("com.acme.MyCustomTokenImpl"))
   {
      byte[] myTokenBytes = tokenHolder.getBytes();

          // Passes these bytes into the constructor of your implementation
          // class for deserialization.
      com.acme.MyCustomTokenImpl myTokenImpl = new com.acme.MyCustomTokenImpl(myTokenBytes);
   }
  }
 }
}

public boolean commit() throws LoginException
{
 // (For more information on what to do during a commit, see
  //   "Developing custom login modules for a system login configuration for JAAS" on page 1521.)
}

// Defines your login module variables
com.ibm.wsspi.security.token.AuthorizationToken customAuthzToken = null;
com.ibm.wsspi.security.token.AuthenticationToken defaultAuthToken = null;
java.util.Map _sharedState = null;
}
```

Configuring outbound identity mapping to a different target realm

By default, when WebSphere Application Server makes an outbound request from one server to another server in a different security realm, the request is rejected. This topic details alternatives for enabling one server to send outbound requests to a target server in a different realm.

About this task

This outbound request is rejected to protect against a rogue server reading potentially sensitive information if successfully impersonating the home of the object. Select one of the following alternative procedures so that one server can send outbound requests to a target server in a different realm. When you are finished with a procedure on the administrative console, click **Apply**.

Procedure

- Do not perform mapping. Instead, allow the existing security information to flow to a trusted target server, even if the target server resides in a different realm. Complete the following steps in the administrative console:
  1. Click **Security > Global security**.
  2. Under RMI/IIOP security, click **CSIv2 outbound authentication**.
  3. Specify the target realms in the **Trusted target realms** field. You can specify each trusted target realm that is separated by a pipe (|) character. For example, specify *server_name*.*domain*:*port_number* for a Lightweight Directory Access Protocol (LDAP) server or the machine name for local operating system. If you want to propagate security attributes to a different target realm, you must specify that target realm in the **Trusted target realms** field.
- Use the Java Authentication and Authorization Service (JAAS) WSLogin application login configuration to create a basic authentication Subject that contains the credentials of the new target realm. This configuration enables you to log in with a realm, user ID, and password that are specific to the user registry of the target realm. You can provide the login information from within the Java Platform, Enterprise Edition (Java EE) application that is making the outbound request or from within the RMI_OUTBOUND system login configuration. These two login options are described in the following information:
  1. Use the WSLogin application login configuration from within the Java EE application to log in and get a Subject that contains the user ID and the password of the target realm. The application can wrap the remote call with a WSSubject.doAs call. For an example, see “Example: Using the WSLogin configuration to create a basic authentication subject” on page 1542.
  2. Use the code sample in “Example: Using the WSLogin configuration to create a basic authentication subject” on page 1542 from this plug point within the RMI_OUTBOUND login configuration. Every outbound Remote Method Invocation (RMI) request passes through this login configuration when it is enabled. Complete the following steps to enable and plug in this login configuration:
     a. Click **Security > Global security**.
     b. Under RMI/IIOP security, click **CSIv2 outbound authentication**.
     c. Select the **Custom outbound mapping** option. If the **Security Attribute Propagation** option is selected, then WebSphere Application Server is already using this login configuration and you do not need to enable custom outbound mapping.
     d. Write a custom login module. For more information, see “Developing custom login modules for a system login configuration for JAAS” on page 1521.

        The “Example: Sample login configuration for RMI_OUTBOUND” on page 1543 shows a custom login module that determines whether the realm names match. In this example, the realm names do not match so the WSLoginmodule is used to create a basic authentication Subject based on custom mapping rules. The custom mapping rules are specific to the customer environment and must be implemented using a realm to user ID and password mapping utility.
     e. Configure the RMI_OUTBOUND login configuration so that your new custom login module is first in the list.
        1) Click **Security > Global security**.
        2) Under Java Authentication and Authorization Service, click **System logins > RMI_OUTBOUND**
        3) Under Additional Properties, click **JAAS login modules > New** to add your login module to the RMI_OUTBOUND configuration.
        4) Return to the JAAS login modules panel for RMI_OUTBOUND.
        5) Click **Set order** to change the order that the login modules are loaded so that your custom login is loaded first.
- Add the use_realm_callback and use_appcontext_callback options to the outbound mapping module for WSLogin. To add these options, complete the following steps:
  1. Click **Security > Global security**.

2. Under Java Authentication and Authorization Service, click **Application logins > WSLogin**.
3. Under Additional properties, click **JAAS login modules > com.ibm.ws.security.common.auth.module.WSLoginModuleImpl**.
4. Under Additional properties, click **Custom Properties > New**.
5. On the Custom properties panel, enter `use_realm_callback` in the **Name** field and `true` in the **Value** field.
6. Click **OK**.
7. Click **New** to enter the second custom property.
8. On the Custom properties panel, enter `use_appcontext_callback` in the **Name** field and `true` in the **Value** field.

The following changes are made to the `security.xml` file:

```
<entries xmi:id="JAASConfigurationEntry_2" alias="WSLogin">
 <loginModules xmi:id="JAASLoginModule_2"
  moduleClassName="com.ibm.ws.security.common.auth.module.proxy.WSLoginModuleProxy"
  authenticationStrategy="REQUIRED">
  <options xmi:id="Property_2" name="delegate"
   value="com.ibm.ws.security.common.auth.module.WSLoginModuleImpl"/>
  <options xmi:id="Property_3" name="use_realm_callback" value="true"/>
  <options xmi:id="Property_4" name="use_appcontext_callback" value="true"/>
 </loginModules>
</entries>
```

Example: Using the WSLogin configuration to create a basic authentication subject:

This example shows how to use the WSLogin application login configuration from within a Java 2 Platform, Enterprise Edition (J2EE) application to log in and get a Subject that contains the user ID and the password of the target realm.

```
javax.security.auth.Subject subject = null;

try
{
  // Create a login context using the WSLogin login configuration and specify a
  // user ID, target realm, and password. Note: If the target_realm_name is the
  // same as the current realm, an authenticated Subject is created. However, if
  // the target_realm_name is different from the current realm, a basic
  // authentication Subject is created that is not validated. This unvalidated
  // Subject is created so that you can send a request to the different target
  // realm with valid security credentials for that realm.
  javax.security.auth.login.LoginContext ctx = new LoginContext("WSLogin",
   new WSCallbackHandlerImpl("userid", "target_realm_name", "password"));

  // Note: The following code is an alternative that validates the user ID and
  // password specified against the target realm. The code performs a remote call
  // to the target server and will return  true if the user ID and password are
  // valid and false if the user ID and password are not valid. If false is
  // returned, a WSLoginFailedException exception is created. You can catch
  // that exception and perform a retry or stop the request from flowing by
  // allowing that exception to surface out of this login.

  // ALTERNATIVE LOGIN CONTEXT THAT VALIDATES THE USER ID AND PASSWORD TO THE
  // TARGET REALM

  /****  currently remarked out ****
  java.util.Map appContext = new java.util.HashMap();
              appContext.put(javax.naming.Context.INITIAL_CONTEXT_FACTORY,
                             "com.ibm.websphere.naming.WsnInitialContextFactory");
              appContext.put(javax.naming.Context.PROVIDER_URL,
                             "corbaloc:iiop:target_host:2809");

  javax.security.auth.login.LoginContext ctx = new LoginContext("WSLogin",
   new WSCallbackHandlerImpl("userid", "target_realm_name", "password", appContext));
  **** currently remarked out  ****/

  // Starts the login
  ctx.login();

  // Gets the Subject from the context
  subject = ctx.getSubject();
 }
 catch (javax.security.auth.login.LoginException e)
 {
  throw new com.ibm.websphere.security.auth.WSLoginFailedException (e.getMessage(), e);
 }

 if (subject != null)
 {
```

```
  // Defines a privileged action that encapsulates your remote request.
java.security.PrivilegedAction myAction = java.security.PrivilegedAction()
  {
   public Object run()
   {
    // Assumes a proxy is already defined. This example method returns a String
    return proxy.remoteRequest();
   }
  });

  // Starts this action using the basic authentication Subject needed for
     // the target realm security requirements.
  String myResult = (String) com.ibm.websphere.security.auth.WSSubject.doAs
        (subject, myAction);
 }
```

Example: Sample login configuration for RMI_OUTBOUND:

This example shows a sample login configuration for RMI_OUTBOUND that determines whether the realm names match between two servers.

```
public customLoginModule()
{
 public void initialize(Subject subject, CallbackHandler callbackHandler,
     Map sharedState, Map options)
 {
     // (For more information on what to do during initialization, see
     // "Developing custom login modules for a system login configuration for JAAS" on page 1521.)
 }

 public boolean login() throws LoginException
 {
     // (For more information on what to do during login, see
     // "Developing custom login modules for a system login configuration for JAAS" on page 1521.)

  // Gets the WSProtocolPolicyCallback object
  Callback callbacks[] = new Callback[1];
   callbacks[0] = new com.ibm.wsspi.security.auth.callback.
          WSProtocolPolicyCallback("Protocol Policy Callback: ");

  try
  {
   callbackHandler.handle(callbacks);
  }
  catch (Exception e)
  {
   // Handles the exception
  }

     // Receives the RMI (CSIv2) policy object for checking the target realm
     // based upon information from the IOR.
     // Note: This object can be used to perform additional security checks.
     // See the application programming interface (API) documentation for
     // more information.
  csiv2PerformPolicy = (CSIv2PerformPolicy) ((WSProtocolPolicyCallback)callbacks[0]).
        getProtocolPolicy();

  // Checks if the realms do not match. If they do not match, then log in to
     // perform a mapping
  if (!csiv2PerformPolicy.getTargetSecurityName().equalsIgnoreCase(csiv2PerformPolicy.
        getCurrentSecurityName()))
  {
   try
   {
    // Do some custom realm -> user ID and password mapping
    MyBasicAuthDataObject myBasicAuthData = MyMappingLogin.lookup
          (csiv2PerformPolicy.getTargetSecurityName());

          // Creates the login context with basic authentication data gathered from
          // custom mapping
    javax.security.auth.login.LoginContext ctx = new LoginContext("WSLogin",
     new WSCallbackHandlerImpl(myBasicAuthData.userid,
       csiv2PerformPolicy.getTargetSecurityName(),
                  myBasicAuthData.password));

     // Starts the login
     ctx.login();

            // Gets the Subject from the context. This subject is used to replace
            // the passed-in Subject during the commit phase.
     basic_auth_subject = ctx.getSubject();
    }
    catch (javax.security.auth.login.LoginException e)
    {
     throw new com.ibm.websphere.security.auth.
               WSLoginFailedException (e.getMessage(), e);
    }
  }
```

```
}

public boolean commit() throws LoginException
{
    // (For more information on what to do during commit, see
    // "Developing custom login modules for a system login configuration for JAAS" on page 1521.)

 if (basic_auth_subject != null)
 {
      // Removes everything from the current Subject and adds everything from the
      // basic_auth_subject
  try
  {
   public final Subject basic_auth_subject_priv = basic_auth_subject;
         // Do this in a doPrivileged code block so that application code
         // does not need to add additional permissions
   java.security.AccessController.doPrivileged(new java.security.
           PrivilegedExceptionAction()
   {
    public Object run() throws WSLoginFailedException
    {
              // Removes everything user-specific from the current outbound
              // Subject. This a temporary Subject for this specific invocation
              // so you are not affecting the Subject set on the thread. You may
              // keep any custom objects that you want to propagate in the Subject.
              // This example removes everything and adds just the new information
              // back in.
     try
     {
      subject.getPublicCredentials().clear();
      subject.getPrivateCredentials().clear();
      subject.getPrincipals().clear();
     }
     catch (Exception e)
     {
      throw new WSLoginFailedException (e.getMessage(), e);
     }

              // Adds everything from basic_auth_subject into the login subject.
              // This completes the mapping to the new user.
     try
     {
      subject.getPublicCredentials().addAll(basic_auth_subject.
                  getPublicCredentials());
      subject.getPrivateCredentials().addAll(basic_auth_subject.
                  getPrivateCredentials());
      subject.getPrincipals().addAll(basic_auth_subject.
                  getPrincipals());
     }
     catch (Exception e)
     {
      throw new WSLoginFailedException (e.getMessage(), e);
     }

     return null;
    }
   });
  }
  catch (PrivilegedActionException e)
  {
   throw new WSLoginFailedException (e.getException().getMessage(),
           e.getException());
  }
 }
}

// Defines your login module variables
com.ibm.wsspi.security.csiv2.CSIv2PerformPolicy csiv2PerformPolicy = null;
javax.security.auth.Subject basic_auth_subject = null;
}
```

Security attribute propagation

With *Security attribute propagation*, WebSphere Application Server can transport security attributes (authenticated Subject contents and security context information) from one server to another in your configuration. WebSphere Application Server might obtain these security attributes from either an enterprise user registry, which queries static attributes, or a custom login module, which can query static or dynamic attributes. Dynamic security attributes, which are custom in nature, might include the authentication strength that is used for the connection, the identity of the original caller, the location of the original caller, the IP address of the original caller, and so on.

Security attribute propagation provides propagation services using Java serialization for any objects that are contained in the Subject. However, Java code must be able to serialize and deserialize these objects.

The Java programming language specifies the rules for how Java code can serialize an object. Because problems can occur when dealing with different platforms and versions of software, WebSphere Application Server also offers a token framework that enables custom serialization functionality. The token framework has other benefits that include the ability to identify the uniqueness of the token. This uniqueness determines how the Subject gets cached and the purpose of the token. The token framework defines four marker token interfaces that enable the WebSphere Application Server runtime to determine how to propagate the token.

Important: Any custom tokens that are used in this framework are not used by WebSphere Application Server for authorization or authentication. The framework serves as a way to notify WebSphere Application Server that you want these tokens propagated in a particular way. WebSphere Application Server handles the propagation details, but does not handle serialization or deserialization of custom tokens. Serialization and deserialization of these custom tokens are carried out by the implementation and handled by a custom login module.

With WebSphere Application Server Version 6.0 and later, a custom Java Authorization Contract for Container (JACC) provider can be configured to enforce access control for Java Platform, Enterprise Edition (Java EE) applications. A custom JACC provider can explore the custom security attributes in the caller JAAS subject in making access control decisions.

When a request is being authenticated, a determination is made by the login modules whether this request is an *initial login* or a *propagation login*. An initial login is the process of authenticating the user information, typically a user ID and password, and then calling the application programming interfaces (APIs) for the remote user registry to look up secure attributes that represent the user access rights. A propagation login is the process of validating the user information, typically a Lightweight Third Party Authentication (LTPA) token, and then deserializing a series of tokens that constitute both custom objects and token framework objects known to WebSphere Application Server.

The following marker tokens are introduced in the framework:

Authorization token

The authorization token contains most of the authorization-related security attributes that are propagated. The default authorization token is used by the WebSphere Application Server authorization engine to make Java Platform, Enterprise Edition (Java EE) authorization decisions. Service providers can use custom authorization token implementations to isolate their data in a different token, perform custom serialization and de-serialization, and make custom authorization decisions using the information in their token at the appropriate time. For information on how to use and implement this token type, see “Using the default propagation token to propagate security attributes” on page 1554 and Implementing a custom propagation token for security attribute propagation.

Single sign-on (SSO) token

A custom SingleSignonToken token that is added to the Subject is automatically added to the response as an HTTP cookie and contains the attributes sent back to web browsers. The token interface getName method with the getVersion method defines the cookie name. WebSphere Application Server defines a default SingleSignonToken token with the LtpaToken name and Version 2. The cookie name added is LtpaToken2. Do not add sensitive information, confidential information, or unencrypted data to the response cookie.

It is also recommended that any time that you use cookies, use the Secure Sockets Layer (SSL) protocol to protect the request. Using an SSO token, web users can authenticate once when accessing web resources across multiple WebSphere Application Servers. A custom SSO token extends this functionality by adding custom processing to the single sign-on scenario. For more information on SSO tokens, see “Implementing single sign-on to minimize web user authentications” on page 1441. For information on how to use and implement this token type, see

"Using the default single sign-on token with default or custom token factory to propagate security attributes" on page 1560 and Implementing a custom single sign-on token for security attribute propagation.

Propagation token

The propagation token is not associated with the authenticated user so it is not stored in the Subject. Instead, the propagation token is stored on the thread and follows the invocation wherever it goes. When a request is sent outbound to another server, the propagation tokens on that thread are sent with the request and the tokens are run by the target server. The attributes that are stored on the thread are propagated regardless of the Java Platform, Enterprise Edition (Java EE) RunAs user switches.

The default propagation token monitors and logs all user switches and host switches. You can add additional information to the default propagation token using the WSSecurityHelper application programming interfaces (APIs). To retrieve and set custom implementations of a propagation token, you can use the WSSecurityPropagationHelper class. For information on how to use and implement this token type, see "Using the default propagation token to propagate security attributes" on page 1554 and Implementing a custom propagation token for security attribute propagation.

Authentication token

The authentication token flows to downstream servers and contains the identity of the user. This token type serves the same function as the Lightweight Third Party Authentication (LTPA) token in previous versions. Although this token type is typically reserved for internal WebSphere Application Server purposes, you can add this token to the Subject and the token is propagated using the getBytes method of the token interface.

A custom authentication token is used solely for the purpose of the service provider that adds it to the Subject. WebSphere Application Server does not use it for authentication purposes because a default authentication token exists that is used for WebSphere Application Server authentication. This token type is available for the service provider to identify how the custom data uses the token to perform custom authentication decisions. For information on how to use and implement this token type, see "Default authentication token" on page 1548 and Implementing a custom authentication token for security attribute propagation .

Kerberos authentication token

The Kerberos authentication token contains Kerberos credentials such as the Kerberos principal name, GSSCredential and Kerberos delegation credential. This token is propagated to the downstream server. Although this token type is typically reserved for internal WebSphere Application Server purposes, if it contains the GSSCredential you can use the getGSSCredential method to extract the GSSCredential. You can then place it in the subject and it can be used for your application. This token is created when you authenticate to WebSphere Application Server with either SPNEGO web or Kerberos authentication.

Horizontal propagation versus downstream propagation

In WebSphere Application Server, both horizontal propagation, which uses single sign-on for web requests, and downstream propagation, which uses Remote Method Invocation over the Internet Inter-ORB Protocol (RMI/IIOP) to access enterprise beans, are available.

Horizontal propagation

In horizontal propagation, security attributes are propagated among front-end servers. The serialized security attributes, which are the Subject contents and the propagation tokens, can contain both static and dynamic attributes. The single sign-on (SSO) token stores additional system-specific information that is needed for horizontal propagation. The information contained in the SSO token tells the receiving server where the originating server is located and how to communicate with that server. Additionally, the SSO token also contains the key to look up the serialized attributes. To enable horizontal propagation, you must

configure the single sign-on token and the web inbound security attribute propagation features. You can configure both of these features using the administrative console.

Figure 1

Performance implications for horizontal propagation

The performance implications of the JMX remote call depends upon your environment. The JMX remote call is used for obtaining the original login attributes. Horizontal propagation reduces many of the remote user registry calls in cases where these calls cause the most performance problems for an application. However, the deserialization of these objects also might cause performance degradation, but this degradation might be less than the remote user registry calls. It is recommended that you test your environment with horizontal propagation enabled and disabled. In cases where you must use horizontal propagation for preserving original login attributes, test whether JMX provides better performance in your environment.

Downstream propagation

In *downstream propagation*, a Subject is generated at the web front-end server, either by a propagation login or a user registry login. WebSphere Application Server propagates the security information downstream for enterprise bean invocations when both Remote Method Invocation (RMI) outbound and inbound propagation are enabled.

Benefits of propagating security attributes

The security attribute propagation feature of WebSphere Application Server has the following benefits:

- Enables WebSphere Application Server to use the security attribute information for authentication and authorization purposes. The propagation of security attributes can eliminate the need for user registry calls at each remote hop along an invocation. Previous versions of WebSphere Application Server propagated only the user name of the authenticated user, but ignored other security attribute information that needed to be regenerated downstream using remote user registry calls. To accentuate the benefits of this new functionality, consider the following example:

  In previous releases, you might use a reverse proxy server (RPSS), such as WebSEAL, to authenticate the user, gather group information, and gather other security attributes. As stated previously, WebSphere Application Server accepted the identity of the authenticated user, but disregarded the additional security attribute information. To create a Java Authentication and Authorization Service (JAAS) Subject containing the needed WSCredential and WSPrincipal objects, WebSphere Application Server made 5 to 6 calls to the user registry. The WSCredential object contains various security information that is required to authorize a Java EE resource. The WSPrincipal object contains the realm name and the user that represents the principal for the Subject.

  In the current release of the Application Server, information that is obtained from the reverse proxy server can be used by WebSphere Application Server and propagated downstream to other server resources without additional calls to the user registry. The retaining of the security attribute information enables you to protect server resources properly by making appropriate authorization and trust-based decisions User switches that occur because of Java EE RunAs configurations do not cause the application server to lose the original caller information. This information is stored in the PropagationToken located on the running thread.
- Enables third-party providers to plug in custom tokens. The token interface contains a getBytes method that enables the token implementation to define custom serialization, encryption methods, or both.
- Provides the ability to have multiple tokens of the same type within a Subject created by different providers. WebSphere Application Server can handle multiple tokens for the same purpose. For example, you might have multiple authorization tokens in the Subject and each token might have distinct authorization attributes that are generated by different providers.

- Provides the ability to have a unique ID for each token type that is used to formulate a more unique subject identifier than just the user name in cases where dynamic attributes might change the context of a user login. The token type has a getUniqueId() method that is used for returning a unique string for caching purposes. For example, you might need to propagate a location ID, which indicates the location from which the user logs into the system. This location ID can be generated during the original login using either an reverse proxy server or the WEB_INBOUND login configuration and added to the Subject prior to serialization. Other attributes might be added to the Subject as well and use a unique ID. All of the unique IDs must be considered for the uniqueness of the entire Subject. WebSphere Application Server has the ability to specify what is unique about the information in the Subject, which might affect how the user accesses the Subject later.

Default authentication token

Do not use the default authentication token in service provider code. This default token is used by the WebSphere Application Server run-time code only and is authentication mechanism specific.

Any modifications to this token by service provider code can potentially cause interoperability problems. If you need to create an authentication token for custom usage, see Implementing a custom authentication token for security attribute propagation for more information.

Changing the token factory that is associated with the default authentication token

When WebSphere Application Server generates a default authentication token, the application server utilizes the TokenFactory class that is specified using the com.ibm.wsspi.security.token.authenticationTokenFactory property. To modify this property using the administrative console, complete the following steps:

1. Click **Security > Global security**.
2. Under Additional properties, click **Custom properties**.

The com.ibm.ws.security.ltpa.LTPATokenFactory token factory is the default for this property. The LTPATokenFactory token factory uses the DESede/ECB/PKCS5Padding cipher. This token factory creates an interoperable Lightweight Third Party Authentication (LTPA) token.

If you associate the com.ibm.ws.security.ltpa.LTPAToken2Factory token factory with the com.ibm.wsspi.security.token.authenticationTokenFactory property, the token is Advanced Encryption Standard (AES) encrypted. However, you need to weigh the performance against your security needs. You might add additional attributes to the authentication token in the Subject during a login that are available downstream.

If you need to perform your own signing and encryption of the default authentication token, you must implement the following classes:

- com.ibm.wsspi.security.ltpa.Token
- com.ibm.wsspi.security.ltpa.TokenFactory

Your token factory implementation instantiates (createToken) and validates (validateTokenBytes) your token implementation. You can use the LTPA keys that are passed into the initialize method of the token factory or you can use your own keys. If you use your own keys, they must be the same everywhere to validate the tokens that are generated using those keys. See the API documentation, available through a link on the front page of the information center, for more information on implementing your own custom token factory. To associate your token factory with the default authentication token using the administrative console, complete the following steps:

1. Click **Security > Global security**.
2. Under Additional properties, click **Custom properties**.
3. Locate the com.ibm.wsspi.security.token.authenticationTokenFactory property and verify that the value of this property matches your custom token factory implementation.

4. Verify that your implementation classes are put into the *install_dir*/classes directory so that the WebSphere Application Server class loader can load the classes.
5. Verify that the QEJBSVR user profile has read, write, and execute (*RWX) authority to the classes directory. You can use the Work with Authority (WRKAUT) command to view the authority permissions for that directory.

Propagating security attributes among application servers

Use the security attribute propagation feature of WebSphere Application Server to send security attribute information regarding the original login to other servers using a token. This topic will help to configure WebSphere Application Server to propagate security attributes to other servers.

About this task

To fully enable security attribute propagation, you must configure the single sign-on (SSO), Common Secure Interoperability Version 2 (CSIv2) inbound, and CSIv2 outbound panels in the WebSphere Application Server administrative console. You can enable just the portions of security attribute propagation relevant to your configuration. For example, you can enable web propagation, which is propagation amongst front-end application servers, using either the push technique (DynaCache) or the pull technique (remote method to originating server).

You also can choose whether to enable Remote Method Invocation (RMI) outbound and inbound propagation, which is commonly called downstream propagation. Typically both types of propagation are enabled for any given cell. In some cases, you might want to choose a different option for a specific application server using the server security panel within the specific application server settings.

Restriction: To prevent propagating the same security attributes among application servers multiple times, WebSphere Application Server verifies that a Lightweight Third Party Authentication (LTPA) token does not exist. Two cases can occur. Absence of the LTPA token tells the Application Server that propagation can proceed. Presence of the LTPA token indicates that propagation has occurred if the LTPA token has been generated within the cluster. However, in the second case, if the LTPA token is present, but has been generated by a server outside the cluster, such as by Tivoli Access Manager, Lotus Domino or a different Application Server cluster, security attributes are not propagated.

To access the server security panel in the administrative console, click **Servers > Application Servers > *server_name***. Under Security, click **Server security**.

Complete the following steps to configure WebSphere Application Server for security attribute propagation:

Procedure

1. Access the WebSphere Application Server administrative console by typing http://*server_name*:*port_number*/ibm/console. The administrative console address might differ if you have previously changed the port number.
2. Click **Security > Global security**.
3. Under Web security, click **Single sign-on (SSO)**.
4. Optional: Select the **Interoperability Mode** option if you need to interoperate with servers that do not support security attribute propagation. Servers that do not support security attribute propagation receive the Lightweight Third Party Authentication (LTPA) token and the Propagation token, but ignore the security attribute information that they do not understand.
5. Select the **Web inbound security attribute propagation** option. The Web inbound security attribute propagation option enables horizontal propagation, which allows the receiving SSO token to retrieve the login information from the original login server. If you do not enable this option, downstream propagation can occur if you enable the Security Attribute Propagation option on both the CSIv2 Inbound authentication and CSIv2 outbound authentication panels.

Typically, you enable the web inbound security attribute propagation option if you need to gather dynamic security attributes set at the original login server that cannot be regenerated at the new front-end server. These attributes include any custom attributes that might be set in the PropagationToken token using the com.ibm.websphere.security.WSSecurityHelper application programming interfaces (APIs). You must determine whether enabling this option improves or degrades the performance of your system. While the option prevents some remote user registry calls, the deserialization and decryption of some tokens might impact performance. In some cases propagation is faster, especially if your user registry is the bottleneck of your topology. It is recommended that you measure the performance of your environment both using and not using this option. When you test the performance, it is recommended that you test in the operating environment of the typical production environment with the typical number of unique users accessing the system simultaneously.

6. Click **Security > Global security**. Under RMI/IIOP security, click **CSIv2 inbound authentication**. The Login configuration field specifies `RMI_INBOUND` as the system login configuration that is used for inbound requests. To add custom Java Authentication and Authorization Service (JAAS) login modules, complete the following steps:
   a. Click **Security > Global security**. Under Java Authentication and Authorization Service, click **System logins**. A list of the system login configurations is displayed. WebSphere Application Server provides the following pre-configured system login configurations: DEFAULT, LTPA, LTPA_WEB, RMI_INBOUND, RMI_OUTBOUND, SWAM, WEB_INBOUND, wssecurity.IDAssertion, and wssecurity.Signature. Do not delete these predefined configurations.

      Note: SWAM is deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release.
   b. Click the name of the login configuration that you want to modify.
   c. Under Additional Properties, click **JAAS Login Modules**. The JAAS Login Modules panel is displayed, which lists all of the login modules that are processed in the login configuration. Do not delete the required JAAS login modules. Instead, you can add custom login modules before or after the required login modules. If you add custom login modules, do not begin their names with com.ibm.ws.security.server.

      You can specify the order in which the login modules are processed by clicking **Set Order**.
7. Select the **Security attribute propagation** option on the CSIv2 inbound authentication panel. When you select **Security Attribute Propagation**, the server advertises to other application servers that it can receive propagated security attributes from another server in the same realm over the Common Secure Interoperability version 2 (CSIv2) protocol.
8. Click **Security > Global security**. Under RMI/IIOP security, click **CSIv2 Outbound authentication**. The CSIv2 outbound authentication panel is displayed. The **Login configuration** field specifies `RMI_OUTBOUND` as the JAAS login configuration that is used for outbound configuration. You cannot change this login configuration. Instead, you can customize this login configuration by completing the substeps that are listed previously for CSIv2 Inbound authentication.
9. Optional: Verify that the **Security Attribute Propagation** option is selected if you want to enable outbound Subject and security context token propagation for the Remote Method Invocation (RMI) protocol. When you select this option, WebSphere Application Server serializes the Subject contents and the PropagationToken contents. After the contents are serialized, the server uses the CSIv2 protocol to send the Subject and PropagationToken token to the target servers that support security attribute propagation. If the receiving server does not support security attribute tokens, WebSphere Application Server sends the Lightweight Third Party Authentication (LTPA) token only.

   Important: WebSphere Application Server propagates only the objects within the Subject that it can serialize. The server propagates custom objects on a best-effort basis.

   When **Security Attribute Propagation** is enabled, WebSphere Application Server adds marker tokens to the Subject to enable the target server to add additional attributes during the inbound login. During the commit phase of the login, the marker tokens and the Subject are marked as read-only and cannot be modified thereafter.

10. Optional: Select the **Custom Outbound Mapping** option if you clear the **Security Attribute Propagation** option and you want to use the RMI_OUTBOUND login configuration. If neither the **Custom Outbound Mapping** option nor the **Security Attribute Propagation** option is selected, WebSphere Application Server does not call the RMI_OUTBOUND login configuration. If you need to plug in a credential mapping login module, you must select the **Custom Outbound Mapping** option.
11. Optional: Specify trusted target realm names in the **Trusted Target Realms** field. By specifying these realm names, information can be sent to servers that reside outside the realm of the sending server to support inbound mapping that is at these downstream servers. To perform outbound mapping to a realm different from the current realm, you must specify the realm in this field so that you can get to this point without having the request rejected because of a realm mismatch. If you need WebSphere Application Server to propagate security attributes to another realm when a request is sent, you must specify the realm name in the **Trusted Target Realms** field. Otherwise, the security attributes are not propagated to the unspecified realm. You can add multiple target realms by adding a pipe (|) delimiter between each entry.
12. Optional: Enable propagation for a pure client. For a pure client to propagate attributes added to the invocation Subject, you must add the following property to the `sas.client.props` file:

    ```
    com.ibm.CSI.rmiOutboundPropagationEnabled=true
    ```

    Note: The `sas.client.props` file is located at `<WAS-HOME>/profiles/<ProfileName>/properties>`.

Results

After completing these steps, you have configured WebSphere Application Server to propagate security attributes to other servers.

What to do next

If you need to disable security attribute propagation, determine whether you need to disable it for either the server level or the cell level.

Attention: Changes to the server-level settings override the cell settings.

To disable security attribute propagation on the server level, complete the following steps:

1. Click **Server > Application Servers > *server_name***.
2. Under Security, click **Server security**.
3. Select the **RMI/IIOP security for this server overrides cell settings** option.
4. Disable security attribute propagation for inbound requests by clicking **CSI inbound authentication** under Additional Properties and clearing the **Security attribute propagation** option.
5. Disable security attribute propagation for outbound requests by clicking **CSI outbound authentication** under Additional Properties and clearing the **Security attribute propagation** option.

To disable security attribute propagation on the cell level, undo each of the steps that you completed to enable security attribute propagation in this task.

Using the default authorization token to propagate security attributes

This topic explains how WebSphere Application Server uses the default authorization token. Consider using the default authorization token when you are looking for a place to add string attributes that get propagated downstream.

About this task

However, make sure that the attributes you add to the authorization token are specific to the user that is associated with the authenticated Subject. If they are not specific to a user, the attributes probably belong in the propagation token, which is also propagated with the request. For more information on the

propagation token, see “Using the default propagation token to propagate security attributes” on page 1554. To add attributes into the authorization token, you must plug in a custom login module into the various system login modules that are configured. Any login module configuration that has the com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule implementation configured can receive propagated information and can generate propagation information that can be sent outbound to another server.

If propagated attributes are not presented to the login configuration during an initial login, a default authorization token is created in the wsMapDefaultInboundLoginModule login module after the login occurs in the ltpaLoginModule login module. You can obtain a reference to the default authorization token from the login method using the sharedState hashmap. You must plug in the custom login module after the wsMapDefaultInboundLoginModule implementation for WebSphere Application Server to see the default authorization token.

For more information on the Java Authentication and Authorization Service (JAAS) programming model, see the Security: Resources for learning article.

Procedure

- Obtain a reference to the default authorization token from the login method.
- Add attributes to the token.
- Read existing attributes used for authorization.
- Add your custom login module to the *profile_root*/classes directory. For more information, see Creating a classes subdirectory in your profile for custom classes.
- Modify the authorization token factory to use a token factory other than the default token factory.

  When WebSphere Application Server generates a default authorization token, the application server utilizes the TokenFactory class that is specified using the com.ibm.wsspi.security.token.authorizationTokenFactory property.

  The com.ibm.ws.security.ltpa.AuthzPropTokenFactory token factory is the default. This token factory encodes the data, but does not encrypt the data in the authorization token. Because the authorization token typically flows over Common Secure Interoperability Version 2 (CSIv2) using Secure Sockets Layer (SSL), encrypting the token is not necessary. However, if you need additional security for the authorization token, you can associate a different token factory implementation with this property to get encryption. For example, if you associate the com.ibm.ws.security.ltpa.LTPAToken2Factory token factory with this property, the token uses Advanced Encryption Standard (AES) encryption. However, you need to weigh the performance impacts against your security needs. Adding sensitive information to the authorization token is one reason to change the token factory implementation to something that encrypts rather than just encodes.

  1. Open the administrative console.
  2. Click **Security > Global security**.
  3. Under Additional properties, click **Custom properties**.
- Perform your own signing and encryption of the default authorization token.

  If you want to perform your own signing and encryption of the default authorization token, you must implement the following classes:

  – com.ibm.wsspi.security.ltpa.Token
  – com.ibm.wsspi.security.ltpa.TokenFactory

  Your token factory implementation instantiates and validates your token implementation. You can use the Lightweight Third Party Authentication (LTPA) keys that are passed into the initialize method of the token factory or you can use your own keys. If you use your own keys, they must be the same everywhere to validate the tokens that are generated using those keys. See the API documentation, that is available through a link on the front page of the information center, for more information on implementing your own custom token factory.
- Associate your token factory with the default authorization token.

To associate your token factory with the default authorization token, using the administrative console, complete the following steps:

1. Click **Security > Global security**.
2. Under Additional properties, click **Custom properties**.
3. Locate the com.ibm.wsspi.security.token.authorizationTokenFactory property and verify that the value of this property matches your custom token factory implementation.
4. Verify that your implementation classes are put into the *app_server_root*/`classes` directory so that the WebSphere Application Server class loader can load the classes.
5. Verify that the QEJBSVR user profile has read, write, and execute (*RWX) authority to the classes directory. You can use the Work with Authority (WRKAUT) command to view the authority permissions for the directory.

Example

The following example shows the complete task of obtaining a reference to the default authorization token from the login method, adding attributes to the token, and reading from the existing attributes that are used for authorization.

```
public customLoginModule()
{
 public void initialize(Subject subject, CallbackHandler callbackHandler,
         Map sharedState, Map options)
 {
     // (For more information on initialization, see
     // "Developing custom login modules for a system login configuration for JAAS" on page 1521.)

  // Get a reference to the sharedState map that is passed in during initialization.
  _sharedState = sharedState;
 }

 public boolean login() throws LoginException
 {
     // (For more information on what to do during login, see
     // "Developing custom login modules for a system login configuration for JAAS" on page 1521.)

  // Look for the default AuthorizationToken in the shared state
  defaultAuthzToken  = (com.ibm.wsspi.security.token.AuthorizationToken)
       sharedState.get
     (com.ibm.wsspi.security.auth.callback.Constants.WSAUTHZTOKEN_KEY);

  // Might not always have one of these generated. It depends on the login
     // configuration setup.
  if (defaultAuthzToken != null)
  {
   try
   {
    // Add a custom attribute
    defaultAuthzToken.addAttribute("key1", "value1");

    // Determine all of the attributes and values that exist in the token.
    java.util.Enumeration listOfAttributes = defaultAuthorizationToken.
              getAttributeNames();

    while (listOfAttributes.hasMoreElements())
    {
     String key = (String) listOfAttributes.nextElement();

     String[] values = (String[]) defaultAuthorizationToken.getAttributes (key);

     for (int i=0;  i<values.length; i++)
     {
      System.out.println ("Key: " + key + ", Value[" + i + "]: "
                 + values[i]);
     }
    }

    // Read the existing uniqueID attribute.
    String[]  uniqueID = defaultAuthzToken.getAttributes
      (com.ibm.wsspi.security.token.AttributeNameConstants.
               WSCREDENTIAL_UNIQUEID);

     // Getthe uniqueID from the String[]
     String unique_id = (uniqueID != null &&
                uniqueID[0] != null) ? uniqueID[0] : "";

    // Read the existing expiration attribute.
    String[]  expiration = defaultAuthzToken.getAttributes
      (com.ibm.wsspi.security.token.AttributeNameConstants.
```

```
            WSCREDENTIAL_EXPIRATION);

     // An example of getting a long expiration value from the string array.
     long expire_time = 0;
     if (expiration != null && expiration[0] != null)
      expire_time = Long.parseLong(expiration[0]);

    // Read the existing display name attribute.
    String[]  securityName = defaultAuthzToken.getAttributes
      (com.ibm.wsspi.security.token.AttributeNameConstants.
            WSCREDENTIAL_SECURITYNAME);

     // Get the display name from the String[]
     String display_name = (securityName != null &&
              securityName[0] != null) ? securityName[0] : "";


    // Read the existing long securityName attribute.
    String[]  longSecurityName = defaultAuthzToken.getAttributes
     (com.ibm.wsspi.security.token.AttributeNameConstants.
            WSCREDENTIAL_LONGSECURITYNAME);

    // Get the long security name from the String[]
    String long_security_name = (longSecurityName != null &&
             longSecurityName[0] != null) ? longSecurityName[0] : "";


    // Read the existing group attribute.
    String[]  groupList = defaultAuthzToken.getAttributes
      (com.ibm.wsspi.security.token.AttributeNameConstants.
            WSCREDENTIAL_GROUPS);

    // Get the groups from the String[]
    ArrayList groups = new ArrayList();
    if (groupList != null)
    {
     for (int i=0; i<groupList.length; i++)
     {
      System.out.println ("group[" + i + "] = " + groupList[i]);
      groups.add(groupList[i]);
     }
    }
   }
   catch (Exception e)
   {
    throw new WSLoginFailedException (e.getMessage(), e);
   }
  }

 }

 public boolean commit() throws LoginException
 {
  // (For more information on what to do during commit, see
     // "Developing custom login modules for a system login configuration for JAAS" on page 1521.)

 }

 private java.util.Map _sharedState = null;
 private com.ibm.wsspi.security.token.AuthorizationToken defaultAuthzToken = null;
}
```

.

Using the default propagation token to propagate security attributes

A default propagation token is located on the running thread for applications and the security infrastructure to use. The product propagates this default propagation token downstream and the token stays on the thread where the invocation lands at each hop.

About this task

The data is available from within the container of any resource where the propagation token lands. Remember that you must enable the propagation feature at each server where a request is sent for propagation to work. Make sure that you enable security attribute propagation for all of the cells in your environment where you want propagation

There is a WSSecurityHelper class that has application programming interfaces (APIs) for accessing the PropagationToken attributes. This topic documents the usage scenarios and includes examples. A close relationship exists between the propagation token and the work area feature. The main difference between

these features is that after you add attributes to the propagation token, you cannot change the attributes. You cannot change these attributes so that the security runtime can add auditable information and have that information remain there for the life of the invocation. Any time that you add an attribute to a specific key, an ArrayList object is stored to hold that attribute. Any new attribute that is added with the same key is added to the ArrayList object. When you call getAttributes, the ArrayList object is converted to a String array and the order is preserved. The first element in the String array is the first attribute added for that specific key.

In the default propagation token, a change flag is kept that logs any data changes to the token. These changes are tracked to enable WebSphere Application Server to know when to send the authentication information downstream again so that the downstream server has those changes. Normally, Common Secure Interoperability Version 2 (CSIv2) maintains a session between servers for an authenticated client. If the propagation token changes, a new session is generated and subsequently a new authentication occurs. Frequent changes to the propagation token during a method cause frequent downstream calls. If you change the token prior to making many downstream calls or you change the token between each downstream call, you might impact security performance.

Procedure

- Obtain the server list from the default propagation token.

  Every time the propagation token is propagated and used to create the authenticated Subject, either horizontally or downstream, the name of the receiving application server is logged into the propagation token. The format of the host is `"Cell:Node:Server"`, which provides you access to the cell name, node name, and server name of each application server that receives the invocation.

  The following code provides you with this list of names and can be called from a Java 2 Platform, Enterprise Edition (J2EE) application.

  The format of each server in the list is: *cell*:*node_name*:*server_name*. The output, for example, is: `myManager:node1:server1`

```
String[] server_list = null;

// If security is disabled on this application server, do not bother checking
if (com.ibm.websphere.security.WSSecurityHelper.isServerSecurityEnabled())
{
 try
 {
  // Gets the server_list string array
  server_list = com.ibm.websphere.security.WSSecurityHelper.getServerList();
 }
 catch (Exception e)
 {
  // Performs normal exception handling for your application
 }

 if (server_list != null)
 {
  // print out each server in the list, server_list[0] is the first server
  for (int i=0; i<server_list.length; i++)
  {
   System.out.println("Server[" + i + "] = " + server_list[i]);
  }
 }
}
```

- Obtain the list of callers, using the getCallerList API.

A default propagation token is generated any time an authenticated user is set on the running thread or anyone tries to add attributes to the propagation token. Whenever an authenticated user is set on the thread, the user is logged in the default propagation token. At times, the same user might be logged in multiple times if the RunAs user is different from the caller. The following list provides the rules that are used to determine if a user that is added to the thread gets logged into the propagation token:

– The current Subject must be authenticated. For example, an unauthenticated Subject is not logged.
– The current authenticated Subject is logged if a Subject is not previously logged.
– The current authenticated Subject is logged if the last authenticated Subject that is logged does not contain the same user.
– The current authenticated Subject is logged on each unique application server that is involved in the propagation process.

The following code sample shows how to use the getCallerList API.

The format of each caller in the list is: *cell*:*node_name*:*server_name*:*realm*:*port_number*/*securityName*. The output, for example, is: `myManager:node1:server1:ldap.austin.ibm.com:389/jsmith`.

```
String[] caller_list = null;

// If security is disabled on this application server, do not check the caller list
if (com.ibm.websphere.security.WSSecurityHelper.isServerSecurityEnabled())
{
 try
 {
  // Gets the caller_list string array
  caller_list = com.ibm.websphere.security.WSSecurityHelper.getCallerList();
 }
 catch (Exception e)
 {
  // Performs normal exception handling for your application
 }

 if (caller_list != null)
 {
  // Prints out each caller in the list, caller_list[0] is the first caller
  for (int i=0; i<caller_list.length;i++)
  {
   System.out.println("Caller[" + i + "] = " + caller_list[i]);
  }
 }
}
```

- Obtain the security name of the first authenticated user, using the getFirst Caller API.

  Whenever you want to know which authenticated caller started the request, you can call the getFirstCaller method and the caller list is parsed. However, this method returns the security name of the caller only. If you need to know more than the security name, call the getCallerList method and retrieve the first entry in the String array. This entry provides all the caller information.

  The following code sample retrieves the security name of the first authenticated caller using the getFirstCaller API.

  The output, for example, is: `jsmith`.

```
String first_caller = null;

// If security is disabled on this application server, do not bother checking
if (com.ibm.websphere.security.WSSecurityHelper.isServerSecurityEnabled())
{
```

```
 try
 {
  // Gets the first caller
  first_caller = com.ibm.websphere.security.WSSecurityHelper.getFirstCaller();

  // Prints out the caller name
  System.out.println("First caller: " + first_caller);
 }
 catch (Exception e)
 {
  // Performs normal exception handling for your application
 }
}
```

- Obtain the name of the first application server for a request, using the getFirstServer method.

  Whenever you want to know what the first application server is for this request, call the getFirstServer method directly.

  The following code sample retrieves the name of the first application server using the getFirstServer API.

  The output, for example, is: `myManager:node1:server1`.

```
String first_server = null;

// If security is disabled on this application server, do not bother checking
if (com.ibm.websphere.security.WSSecurityHelper.isServerSecurityEnabled())
{
 try
 {
  // Gets the first server
  first_server = com.ibm.websphere.security.WSSecurityHelper.getFirstServer();

  // Prints out the server name
  System.out.println("First server: " + first_server);
 }
 catch (Exception e)
 {
  // Performs normal exception handling for your application
 }
}
```

- Add custom attributes to the default propagation token, using the addPropagationAttribute API.

  You can add custom attributes to the default propagation token for application usage. This token follows the request downstream so that the attributes are available when needed. When you use the default propagation token to add attributes, you must understand the following issues:

  – Adding information to the propagation token affects CSIv2 session caching. Add information sparingly between remote requests.
  – After you add information with a specific key, the information cannot be removed.
  – You can add as many values to a specific key as you need. However, all of the values must be available from a returned String array in the order that they were added.
  – The propagation token is available only on servers where propagation and security are enabled.
  – The Java 2 Security javax.security.auth.AuthPermission wssecurity.addPropagationAttribute attribute is needed to add attributes to the default propagation token.
  – An application cannot use keys that begin with either com.ibm.websphere.security or com.ibm.wsspi.security. These prefixes are reserved for system usage.

  The following code sample shows how to use the addPropagationAttribute API.

```
// If security is disabled on this application server,
    // do not check the status of server security
if (com.ibm.websphere.security.WSSecurityHelper.isServerSecurityEnabled())
{
 try
 {
  // Specifies the key and values
  String key = "mykey";
  String value1 = "value1";
  String value2 = "value2";

  // Sets key, value1
      com.ibm.websphere.security.WSSecurityHelper.
      addPropagationAttribute (key, value1);

  // Sets key, value2
  String[] previous_values = com.ibm.websphere.security.WSSecurityHelper.
         addPropagationAttribute (key, value2);

  // Note: previous_values should contain value1
 }
 catch (Exception e)
 {
  // Performs normal exception handling for your application
 }
}
```

- Obtain your custom attributes with the get PropagationAttributes API.

  Custom attributes are added to the default propagation token using the addPropagationAttribute API. Retrieve these attributes using the getPropagationAttributes API. This token follows the request downstream so the attributes are available when needed. When you use the default propagation token to retrieve attributes, you must understand the following issues:

  – The propagation token is available only on servers where propagation and security are enabled.
  – The Java 2 Security javax.security.auth.AuthPermission "wssecurity.getPropagationAttributes" permission is needed to retrieve attributes from the default propagation token.

  See Adding custom attributes to the default PropagationToken to add attributes using the addPropagationAttributes API.

  The following code sample shows how to use the getPropagationAttributes API.

```
// If security is disabled on this application server, do not bother checking
if (com.ibm.websphere.security.WSSecurityHelper.isServerSecurityEnabled())
{
 try
 {
  String key = "mykey";
  String[] values = null;

  // Sets key, value1
      values = com.ibm.websphere.security.WSSecurityHelper.
      getPropagationAttributes (key);

  // Prints the values
  for (int i=0; i<values.length; i++)
  {
   System.out.println("Value[" + i + "] = " + values[i]);
```

```
   }
  }
  catch (Exception e)
  {
   // Performs normal exception handling for your application
  }
 }
```

The output, for example, is:

```
Value[0] = value1
Value[1] = value2
```

- Modify the propagation token factory configuration to use a token factory other than the default token factory.

  When WebSphere Application Server generates a default propagation token, the Application Server utilizes the TokenFactory class that is specified using the com.ibm.wsspi.security.token.propagationTokenFactory property.

  The default token factory that is specified for this property is called com.ibm.ws.security.ltpa.AuthzPropTokenFactory. This token factory encodes the data in the propagation token and does not encrypt the data. Because the propagation token typically flows over CSIv2 using Secure Sockets Layer (SSL), encrypting the token is not required. However, if you need additional security for the propagation token, you can associate a different token factory implementation with this property to get encryption. For example, if you choose to associate the com.ibm.ws.security.ltpa.LTPAToken2Factory token factory with this property, the token is AES encrypted. However, you need to weigh the performance impacts against your security needs. Adding sensitive information to the propagation token is a good reason to change the token factory implementation to something that encrypts rather than just encodes.

  1. Open the administrative console.
  2. Click **Security > Global security**.
  3. Click **Custom properties**.

- Perform your own signing and encryption of the default propagation token.

  If you want to perform your own signing and encryption of the default propagation token, you must implement the following classes:

  – com.ibm.wsspi.security.ltpa.Token
  – com.ibm.wsspi.security.ltpa.TokenFactory

  Your token factory implementation instantiates and validates your token implementation. You can choose to use the Lightweight Third Party Authentication (LTPA) keys and have them pass into the initialize method of the token factory, or you can use your own keys. If you use your own keys, they must be the same everywhere to validate the tokens that are generated using those keys. See the API documentation, available through a link on the front page of the information center, for more information on implementing your own custom token factory.

- Associate your token factory with the default propagation token.

  1. Open the administrative console.
  2. Click **Security > Global security**.
  3. Click **Custom properties**.
  4. Locate the com.ibm.wsspi.security.token.propagationTokenFactory property and verify that the value of this property matches your custom token factory implementation.
  5. Verify that your implementation classes are put into the *app_server_root*/`classes` directory so that the WebSphere Application Server class loader can load the classes.
  6. Verify that the QEJBSVR user profile has read, write, and execute (*RWX) authority to the classes directory. You can use the Work with Authority (WRKAUT) command to view the authority permissions for the directory.

Example

Using the default single sign-on token with default or custom token factory to propagate security attributes

Do not use the default single sign-on token in service provider code. This default token is used by the WebSphere Application Server run-time code only.

Before you begin

Size limitations exist for this token when it is added as an HTTP cookie. If you need to create an HTTP cookie using this token framework, you can implement a custom single sign-on token. To implement a custom single sign-on token see Implementing a custom single sign-on token for security attribute propagation for more information.

Procedure

- Modify the single sign-on token factory configuration to use a token factory other than the default token factory.

  When the default single sign-on token is generated, the application server utilizes the TokenFactory class that is specified using the com.ibm.wsspi.security.token.singleSignonTokenFactory property. Use the administrative console to modify the property.

  The com.ibm.ws.security.ltpa.LTPAToken2Factory token factory is the default that is specified for this property. This token factory creates a single sign-on (SSO) token called LtpaToken2, which WebSphere Application Server uses for propagation. This token factory uses the AES/CBC/PKCS5Padding cipher.

  1. Open the administrative console.
  2. Click **Security > Global security**.
  3. Under Authentication, click **Custom properties**.
- Perform your own signing and encryption of the default single sign-on token.

  If you need to perform your own signing and encryption of the default single sign-on token, you must implement the following classes:

  – com.ibm.wsspi.security.ltpa.Token

  – com.ibm.wsspi.security.ltpa.TokenFactory

  Your token factory implementation instantiates (createToken) and validates (validateTokenBytes) your token implementation. You can use the Lightweight Third-Party Authentication (LTPA) keys passed into the initialize method of the token factory or you can use your own keys. If you use your own keys, they must be the same everywhere to validate the tokens that are generated using those keys. See the API reference information for more information on implementing your own custom token factory.
- Associate your own token factory with the default single sign-on token.
  1. Open the administrative console.
  2. Click **Security > Global security**.
  3. Under Authentication, click **Custom properties**.
  4. Locate the com.ibm.wsspi.security.token.singleSignonTokenFactory property and verify that the value of this property matches your custom TokenFactory implementation.
  5. Verify that your implementation classes are put into the *app_server_root*/`classes` directory so that the WebSphere Application Server class loader can load the classes.
  6. Verify that the QEJBSVR user profile has read, write, and execute (*RWX) authority to the classes directory. You can use the Work with Authority (WRKAUT) command to view the authority permissions for the directory.

Configuring the authentication cache

The security authentication cache affects the frequency of rehashing and the distribution of the hash algorithms.

About this task

To configure the authentication cache properties, complete the following steps:

Procedure

1. Click **Servers > Application Servers > *server_name*** .
2. Under Server infrastructure, click **Java and Process Management > Process definition**.
3. Under Additional properties, click **Java Virtual Machine > Custom Properties**.
4. Click **New** to specify a new custom property.

What to do next

For information on the supported authentication cache properties, see “Authentication cache settings” on page 1252.

Configuring Common Secure Interoperability Version 2 (CSIV2) inbound and outbound communication settings

WebSphere Application Server enables you to specify Internet Inter-ORB Protocol (IIOP) authentication for both inbound and outbound authentication requests. For inbound requests, you can specify the type of accepted authentication, such as basic authentication. For outbound requests, you can specify properties such as type of authentication, identity assertion or login configurations that are used for requests to downstream servers.

About this task

Complete the following steps to configure Common Secure Interoperability Version 2 (CSIV2) and Security Authentication Service (SAS).

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Procedure

1. Determine how to configure security inbound and outbound at each point in your infrastructure.

   For example, you might have a Java client communicating with an Enterprise JavaBeans (EJB) application server, which in turn communicates to a downstream EJB application server.

   The Java client utilizes the `sas.client.props` file to configure outbound security. Pure clients must configure outbound security only.

   The upstream EJB application server configures inbound security to handle the correct type of authentication from the Java client. The upstream EJB application server utilizes the outbound security configuration when going to the downstream EJB application server.

   This type of authentication might be different than what you expect from the Java client into the upstream EJB application server. Security might be tighter between the pure client and the first EJB server, depending on your infrastructure. The downstream EJB server utilizes the inbound security configuration to accept requests from the upstream EJB server. These two servers require similar configuration options as well. If the downstream EJB application server communicates to other downstream servers, the outbound security might require a special configuration.
2. Specify the type of authentication.

   By default, authentication by a user ID and password is performed.

   Both Java client certificate authentication and identity assertion are disabled by default. If you want this type of authentication performed at every tier, use the CSIv2 authentication protocol configuration as is. However, if you have any special requirements where some servers authenticate differently from other servers, consider how to configure CSIv2 to its best advantage.

3. Configure clients and servers.

   Configuring a pure Java client is done through the `sas.client.props` file, where properties are modified.

   Configuring servers is always done from the administrative console or scripting, either from the security navigation for cell-level configurations or from the server security of the application server for server-level configurations. If you want some servers to authenticate differently from others, modify some of the server-level configurations. When you modify the server-level configurations, you are overriding the cell-level configurations.

What to do next

Use CSIV2 inbound communications settings for configuring the type of authentication information that is contained in an incoming request or transport.

Use CSIV2 outbound communications settings to specify the features that a server supports when acting as a client to another downstream server.

Configuring Common Secure Interoperability Version 2 inbound communications

Inbound communications refers to the configuration that determines the type of accepted authentication for inbound requests. This authentication is advertised in the interoperable object reference (IOR) that the client retrieves from the name server.

Procedure

1. Start the administrative console.
2. Click **Security > Global security**.
3. Under RMI/IIOP security, click **CSIv2 inbound communications**.
4. Consider the following layers of security:
   - Identity assertion (attribute layer).

     When selected, this server accepts identity tokens from upstream servers. If the server receives an identity token, the identity is taken from an originating client. For example, the identity is in the same form that the originating client presented to the first server. An upstream server sends the identity of the originating client. The format of the identity can be either a principal name, a distinguished name, or a certificate chain. In some cases, the identity is anonymous. It is important to trust the upstream server that sends the identity token because the identity authenticates on this server. Trust of the upstream server is established either using Secure Sockets Layer (SSL) client certificate authentication or basic authentication. You must select one of the two layers of authentication in both inbound and outbound authentication when you choose identity assertion.

     The server ID is sent in the client authentication token with the identity token. The server ID is checked against the trusted server ID list. If the server ID is on the trusted server list, the server ID is authenticated. If the server ID is valid, the identity token is put into a credential and used for authorization of the request.

     Note: When identity assertion is enabled, message layer or transport layer should be enabled also. For server-to-server communication, besides enabling transport layer/client authentication, identity assertion or message layer should be enabled also.

     For more information, refer to Identity assertion.
   - Message layer:

     Basic authentication (GSSUP):

     This type of authentication is the most typical. The user ID and password or authenticated token is sent from a pure client or from an upstream server. When a user ID and password are received at the server, they are authenticated with the user registry of the downstream server.

     Lightweight Third Party Authentication (LTPA):

In this case, an LTPA token is sent from the upstream server. Note that if you choose LTPA, then both servers must share the same LTPA keys

Kerberos (KRB5):

To select Kerberos, the active authentication mechanism must be Kerberos. In this case, a Kerberos token is sent from the upstream server.

For more information, read about Message layer authentication.

- Secure Sockets Layer client certificate authentication (transport layer).

  The SSL client certificate is used to authenticate instead of using user ID and Password. If a server delegates an identity to a downstream server, the identity comes from either the message layer (a client authentication token) or the attribute layer (an identity token), and not from the transport layer through the client certificate authentication.

  A client has an SSL client certificate that is stored in the keystore file of the client configuration. When SSL client authentication is enabled on this server, the server requests that the client send the SSL client certificate when the connection is established. The certificate chain is available on the socket whenever a request is sent to the server. The server request interceptor gets the certificate chain from the socket and maps this certificate chain to a user in the user registry. This type of authentication is optimal for communicating directly from a client to a server. However, when you have to go downstream, the identity typically flows over the message layer or through identity assertion.

5. Consider the following points when deciding what type of authentication to accept:
   - A server can receive multiple layers simultaneously, so an order of precedence rule decides which identity to use. The identity assertion layer has the highest priority, the message layer follows, and the transport layer has the lowest priority. The SSL client certificate authentication is used when it is the only layer provided. If the message layer and the transport layer are provided, the message layer is used to establish the identity for authorization. The identity assertion layer is used to establish precedence when provided.
   - Does this server usually receive requests from a client, from a server, or both? If the server always receives requests from a client, identity assertion is not needed. You can choose either the message layer, the transport layer, or both. You also can decide when authentication is required or just supported. To select a layer as required, the sending client must supply this layer, or the request is rejected. However, if the layer is only supported, the layer might not be supplied.
   - What kind of client identity is supplied? If the client identity is client certificates authentication and you want the certificate chain to flow downstream so that it maps to the downstream server user registries, identity assertion is the appropriate choice. Identity assertion preserves the format of the originating client. If the originating client authenticated with a user ID and password, a principal identity is sent. If authentication is done with a certificate, the certificate chain is sent.

     In some cases, if the client authenticated with a token and a Lightweight Directory Access Protocol (LDAP) server is the user registry, then a distinguished name (DN) is sent.
6. Configure a trusted server list. When identity assertion is selected for inbound requests, insert a pipe-separated (|) list of server administrator IDs to which this server can support identity token submission. For backwards compatibility, you can still use a comma-delimited list. However, if the server ID is a distinguished name (DN), then you must use a pipe-delimited (|) list because a comma delimiter does not work. If you choose to support any server sending an identity token, you can enter an asterisk (*) in this field. This action is called *presumed trust*. In this case, use SSL client certificate authentication between servers to establish the trust.
7. Configure session management. You can choose either *stateful* or *stateless* security. Performance is optimum when choosing stateful sessions. The first method request between a client and server is authenticated. All subsequent requests (or until the credential token expires) reuse the session information, including the credential. A client sends a context ID for subsequent requests. The context ID is scoped to the connection for uniqueness.

Results

When you finish configuring this panel, you have configured most of the information that a client gathers when determining what to send to this server. A client or server outbound configuration with this server inbound configuration, determines the security that is applied. When you know what clients send, the configuration is simple. However, if you have a diverse set of clients with differing security requirements, your server considers various layers of authentication.

For a J2EE application server, the authentication choice is usually either identity assertion or message layer because you want the identity of the originating client delegated downstream. You cannot easily delegate a client certificate using an SSL connection. It is acceptable to enable the transport layer because additional server security, as the additional client certificate portion of the SSL handshake, adds some overhead to the overall SSL connection establishment.

What to do next

After you determine which type of authentication data this server might receive, you can determine what to select for outbound security. For more information, see Configuring Common Secure Interoperability Version 2 outbound authentication.

Common Secure Interoperability Version 2 inbound communications settings:

Use this page to specify the features that a server supports for a client accessing its resources.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. From Authentication, click **RMI/IIOP security > CSIv2 inbound communications**.

Use common secure interoperability (CSI) inbound communications settings for configuring the type of authentication information that is contained in an incoming request or transport.

Authentication features include three layers of authentication that you can use simultaneously:

- **CSIv2 attribute layer**. The attribute layer might contain an identity token, which is an identity from an upstream server that already is authenticated. The identity layer has the highest priority, followed by the message layer, and then the transport layer. If a client sends all three, only the identity layer is used. The only way to use the SSL client certificate as the identity is if it is the only information that is presented during the request. The client picks up the interoperable object reference (IOR) from the namespace and reads the values from the tagged component to determine what the server needs for security.
- **CSIv2 transport layer**. The transport layer, which is the lowest layer, might contain a Secure Sockets Layer (SSL) client certificate as the identity.
- **CSIv2 message layer**. The message layer might contain a user ID and password or an authenticated token with an expiration.

Propagate security attributes:

Specifies support for security attribute propagation during login requests. When you select this option, the application server retains additional information about the login request, such as the authentication strength used, and retains the identity and location of the request originator.

If you do not select this option, the application server does not accept any additional login information to propagate to downstream servers.

Default: Enabled

Important: When you use the replication services, ensure that the **Propagate security attributes** option is enabled.

Use identity assertion:

Specifies that identity assertion is a way to assert identities from one server to another during a downstream Enterprise JavaBeans (EJB) invocation.

This server does not authenticate the asserted identity again because it trusts the upstream server. Identity assertion takes precedence over all other types of authentication.

Identity assertion is performed in the attribute layer and is only applicable on servers. The principal determined at the server is based on precedence rules. If identity assertion is used, the identity is always derived from the attribute layer. If basic authentication is used without identity assertion, the identity is always derived from the message layer. Finally, if SSL client certificate authentication is performed without either basic authentication, or identity assertion, then the identity is derived from the transport layer.

The identity asserted is the invocation credential that is determined by the RunAs mode for the enterprise bean. If the RunAs mode is Client, the identity is the client identity. If the RunAs mode is System, the identity is the server identity. If the RunAs mode is Specified, the identity is the one specified. The receiving server receives the identity in an identity token and also receives the sending server identity in a client authentication token. The receiving server validates the sending server identity as a trusted identity through the Trusted Server IDs entry box. Enter a list of pipe-separated (|) principal names, for example, `serverid1|serverid2|serverid3`.

All identity token types map to the user ID field of the active user registry. For an ITTPrincipal identity token, this token maps one-to-one with the user ID fields. For an ITTDistinguishedName identity token, the value from the first equal sign is mapped to the user ID field. For an ITTCertChain identity token, the value from the first equal sign of the distinguished name is mapped to the user ID field.

When authenticating to an LDAP user registry, the LDAP filters determine how an identity of type ITTCertChain and ITTDistinguishedName get mapped to the registry. If the token type is ITTPrincipal, then the principal gets mapped to the UID field in the LDAP registry.

Default: Disabled

Trusted identities:

Specifies the trusted identity that is sent from the sending server to the receiving server.

Specifies a pipe-separated (|) list of trusted server administrator user IDs, which are trusted to perform identity assertion to this server. For example, `serverid1|serverid2|serverid3`. The application server supports the comma (,) character as the list delimiter for backwards compatibility. The application server checks the comma character when the pipe character (|) fails to find a valid trusted server ID.

Use this list to decide whether a server is trusted. Even if the server is on the list, the sending server must still authenticate with the receiving server to accept the identity token of the sending server.

Data type: String

Client certificate authentication:

Specifies that authentication occurs when the initial connection is made between the client and the server during a method request.

In the transport layer, Secure Sockets Layer (SSL) client certificate authentication occurs. In the message layer, basic authentication (user ID and password) is used. Client certificate authentication typically performs better than message layer authentication, but requires some additional setup. These additional steps involve verifying that the server trusts the signer certificate of each client to which it is connected. If the client uses a certificate authority (CA) to create its personal certificate, you only need the CA root certificate in the server signer section of the SSL trust file.

When the certificate is authenticated to a Lightweight Directory Access Protocol (LDAP) user registry, the distinguished name (DN) is mapped based on the filter that is specified when configuring LDAP. When the certificate is authenticated to a local OS user registry, the first attribute of the distinguished name (DN) in the certificate, which is typically the common name, is mapped to the user ID in the registry.

The identity from client certificates is used only if no other layer of authentication is presented to the server.

Never Specifies that clients cannot attempt Secure Sockets Layer (SSL) client certificate authentication with this server.

Supported
Specifies that clients connecting to this server can authenticate using SSL client certificates. However, the server can invoke a method without this type of authentication. For example, anonymous or basic authentication can be used instead.

Required
Specifies that clients connecting to this server must authenticate using SSL client certificates before invoking the method.

Transport:

Specifies whether client processes connect to the server using one of its connected transports.

You can choose either Secure Sockets Layer (SSL), TCP/IP or both as the inbound transport that a server supports. If you specify TCP/IP, the server only supports TCP/IP and cannot accept SSL connections. If you specify SSL-supported, this server can support either TCP/IP or SSL connections. If you specify SSL-required, then any server communicating with this one must use SSL.

Note: This option is not available on the z/OS platform unless both Version 6.1 and earlier nodes exist in the cell.

TCP/IP
If you select **TCP/IP**, then the server opens a TCP/IP listener port only and all inbound requests do not have SSL protection.

SSL-required
If you select **SSL-required**, then the server opens an SSL listener port only and all inbound requests are received using SSL.

SSL-supported
If you select **SSL-supported**, then the server opens both a TCP/IP and an SSL listener port and most inbound requests are received using SSL.

Provide a fixed port number for the following ports. A zero port number indicates that a dynamic assignment is made at run time.

CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS

Default: SSL-Supported

Range: TCP/IP, SSL Required, SSL-Supported

SSL settings:

Specifies a list of predefined SSL settings to choose from for inbound connection.

Data type: String
Default: DefaultSSLSettings
DefaultIIOPSSL
Range: Any SSL settings configured in the SSL Configuration Repertoire

Message layer authentication:

The following options are available for message layer authentication:

Never Specifies that this server cannot accept authentication using any of the mechanisms selected below.

Supported
Specifies that a client communicating with this server can authenticate using any of the mechanisms selected below. However, a method might be invoked without this type of authentication. For example, an anonymous or client certificate might be used instead.

Required
Specifies that clients communicating with this server must specify authentication information using of the mechanisms selected below for any method request.

Allow client to server authentication with::

Specifies client-to-server authentication using Kerberos, LTPA or Basic authentication.

The following options are available for client to server authentication:

Kerberos (KRB5)
Select to specify Kerberos as the authentication mechanism. You must first configure the Kerberos authentication mechanism. Read about Configuring Kerberos as the authentication mechanism using the administrative console for more information.

LTPA Select to specify the LTPA token authentication

Basic authentication
Basic authentication is Generic Security Services Username Password (GSSUP). This type of authentication typically involves sending a user ID and a password from the client to the server for authentication.

If you select **Basic Authentication** and **LTPA**, and the active authentication mechanism is LTPA, a user name, password, and LTPA tokens are accepted.

If you select **Basic Authentication** and **KRB5** and the active authentication mechanism is KRB5, a user name, password, Kerberos token and LTPA tokens are accepted.

If you do not select **Basic Authentication**, a user name and password are not accepted by the server.

Login configuration:

Specifies the type of system login configuration to use for inbound authentication.

You can add custom login modules by clicking **Security > Global security**. From Authentication, click **Java Authentication and Authorization Service > System logins**.

Stateful sessions:

Select this option to enable stateful sessions, which are used mostly for performance improvements.

The first contact between a client and server must fully authenticate. However, all subsequent contacts with valid sessions reuse the security information. The client passes a context ID to the server, and the ID is used to look up the session. The context ID is scoped to the connection, which guarantees uniqueness. Whenever the security session is not valid and the authentication retry is enabled, which is the default, the client-side security interceptor invalidates the client-side session and submits the request again without user awareness. This situation might occur if the session does not exist on the server; for example, the server failed and resumed operation. When this value is disabled, each method invocation must authenticate again.

Default: Enabled

Trusted authentication realms - inbound:

Select this link to establish inbound trust for realms. Inbound authentication realm settings are not specific to CSIv2; you can also configure which realms to grant inbound trust to for multiple security domains.

Inbound authentication refers to the configuration that determines the type of accepted authentication for inbound requests. This authentication is advertised in the interoperable object reference (IOR) that the client retrieves from the name server.

Configuring Common Secure Interoperability Version 2 outbound communications

The following choices are available when configuring the Common Secure Interoperability Version 2 (CSIv2) outbound communications panel.

Before you begin

Outbound communications refers to the configuration that determines the type of authentication that is performed for outbound requests to downstream servers. Several *layers* or *methods* of authentication can occur. The downstream server inbound authentication configuration must support at least one choice made in this server outbound authentication configuration. If nothing is supported, the request might go outbound as unauthenticated. This situation does not create a security problem because the authorization runtime is responsible for preventing access to protected resources. However, if you choose to prevent an unauthenticated credential from going outbound, you might want to designate one of the authentication layers as required, rather than supported. If a downstream server does not support authentication, then when authentication is required, the method request fails to go outbound.

About this task

The following choices are available in the Common Secure Interoperability Version 2 (CSIv2) outbound communications panel. Remember that you are not required to complete these steps in the displayed order. Rather, these steps are provided to help you understand your choices for configuring outbound communications.

Procedure

- Select **Identity Assertion** (attribute layer). When selected, this server sends an identity token to a downstream server if the downstream server supports identity assertion. When an originating client authenticates to this server, the authentication information supplied is preserved in the outbound identity token. If the client authenticating to this server uses client certificate authentication, then the identity

token format is a certificate chain, containing the exact client certificate chain from the inbound socket. The same scenario is true for other mechanisms of authentication. Read theIdentity Assertion topic for more information.

- Select **User ID** and **Password** (message layer). This type of authentication is the most typical. The user ID and password (if BasicAuth credential) or authenticated token (if authenticated credential) are sent outbound to the downstream server if the downstream server supports message layer authentication in the inbound authentication panel. Refer to the Message Layer Authentication article for more information.
- Select **SSL Client certificate authentication** (transport layer). The main reason to enable outbound Secure Sockets Layer (SSL) client authentication from one server to a downstream server is to create a trusted environment between those servers. For delegating client credentials, use one of the two layers mentioned previously. However, you might want to create SSL personal certificates for all the servers in your domain, and only trust those servers in your SSL truststore file. No other servers or clients can connect to the servers in your domain, except at the tiers where you want them. This process can protect your enterprise bean servers from access by anything other than your servlet servers.

Example

Typically, the outbound authentication configuration is for an upstream server to communicate with a downstream server. Most likely, the upstream server is a servlet server and the downstream server is an Enterprise JavaBeans (EJB) server. On a servlet server, the client authentication that is performed to access the servlet can be one of many different types of authentication, including client certificate and basic authentication. When receiving basic authentication data, whether through a prompt login or a form-based login, the basic authentication information is typically authenticated to from a credential of the mechanism type that is supported by the server, such as the Lightweight Third Party Authentication (LTPA). When LTPA is the mechanism, a forwardable token exists in the credential. Choose the message layer (BasicAuth) authentication to propagate the client credentials. If the credential is created using a certificate login and you want to preserve sending the certificate downstream, you might decide to go outbound with identity assertion.

Common Secure Interoperability Version 2 outbound communications settings:

Use this page to specify the features that a server supports when acting as a client to another downstream server.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. From Authentication, click **RMI/IIOP security > CSIv2 outbound communications**.

Authentication features include three layers of authentication that you can use simultaneously:

- **CSIv2 attribute layer**. The attribute layer might contain an identity token, which is an identity from an upstream server that already is authenticated. The identity layer has the highest priority, followed by the message layer, and then the transport layer. If a client sends all three, only the identity layer is used. The only way to use the SSL client certificate as the identity is if it is the only information that is presented during the request. The client picks up the interoperable object reference (IOR) from the namespace and reads the values from the tagged component to determine what the server needs for security.
- **CSIv2 transport layer**. The transport layer, which is the lowest layer, might contain a Secure Sockets Layer (SSL) client certificate as the identity.
- **CSIv2 message layer**. The message layer might contain a user ID and password or an authenticated token with an expiration.

Propagate security attributes:

Specifies to support security attribute propagation during login requests. When you select this option, the application server retains additional information about the login request, such as the authentication strength used, and retains the identity and location of the request originator.

If you do not select this option, the application server does not accept any additional login information to propagate to downstream servers.

Default: Enabled

Important: When you use the replication services, ensure that the **Propagate security attributes** option is enabled.

Use identity assertion:

Specifies that identity assertion is a way to assert identities from one server to another during a downstream Enterprise JavaBeans (EJB) invocation.

This server does not authenticate the asserted identity again because it trusts the upstream server. Identity assertion takes precedence over all other types of authentication.

Identity assertion is performed in the attribute layer and is only applicable on servers. The principal determined at the server is based on precedence rules. If identity assertion is performed, the identity is always derived from the attribute layer. If basic authentication is used without identity assertion, the identity is always derived from the message layer. Finally, if SSL client certificate authentication is performed without either basic authentication, or identity assertion, then the identity is derived from the transport layer.

The identity asserted is the invocation credential that is determined by the RunAs mode for the enterprise bean. If the RunAs mode is Client, the identity is the client identity. If the RunAs mode is System, the identity is the server identity. If the RunAs mode is Specified, the identity is the one specified. The receiving server receives the identity in an identity token and also receives the sending server identity in a client authentication token. The receiving server validates the sending server identity as a trusted identity through the Trusted Server IDs entry box. Enter a list of pipe-separated (|) principal names, for example, `serverid1|serverid2|serverid3`.

All identity token types map to the user ID field of the active user registry. For an ITTPrincipal identity token, this token maps one-to-one with the user ID fields. For an ITTDistinguishedName identity token, the value from the first equal sign is mapped to the user ID field. For an ITTCertChain identity token, the value from the first equal sign of the distinguished name is mapped to the user ID field.

When authenticating to an LDAP user registry, the LDAP filters determine how an identity of type ITTCertChain and ITTDistinguishedName get mapped to the registry. If the token type is ITTPrincipal, then the principal gets mapped to the UID field in the LDAP registry.

Default: Disabled

Use server-trusted identity:

Specifies the server identity that the application server uses to establish trust with the target server. The server identity can be sent using one of the following methods:

- A server ID and password when the server password is specified in the registry configuration.
- A server ID in a Lightweight Third Party Authentication (LTPA) token when the internal server ID is used.

For interoperability with application servers other than WebSphere Application Server, use one of the following methods:

- Configure the server ID and password in the registry.
- Select the **Server-trusted identity** option and specify the trusted identity and password so that an interoperable Generic Security Services Username Password (GSSUP) token is sent instead of an LTPA token.

Default: Disabled

Specify an alternative trusted identity:

Specifies an alternative user as the trusted identity that is sent to the target servers instead of sending the server identity.

This option is recommended for identity assertion. The identity is automatically trusted when it is sent within the same cell and does not need to be in the trusted identities list within the same cell. However, this identity must be in the registry of the target servers in an external cell, and the user ID must be on the trusted identities list or the identity is rejected during trust evaluation.

Note: You must select Basic Authentication under the Message Layer authentication section to send an alternative trusted identity. If you do not select Basic Authentication, then choose the Server Identity instead.

Default: Disabled

Trusted identity:

Specifies the trusted identity that is sent from the sending server to the receiving server.

If you specify an identity in this field, it can be selected on the panel for your configured user account repository. If you do not specify an identity, a Lightweight Third Party Authentication (LTPA) token is sent between the servers.

Specifies a pipe-separated (|) list of trusted server administrator user IDs, which are trusted to perform identity assertion to this server. For example, `serverid1|serverid2|serverid3`. The application server supports the comma (,) character as the list delimiter for backwards compatibility. The application server checks the comma character when the pipe character (|) fails to find a valid trusted server ID.

Use this list to decide whether a server is trusted. Even if the server is on the list, the sending server must still authenticate with the receiving server to accept the identity token of the sending server.

Password:

Specifies the password that is associated with the trusted identity.

Data type: Text

Confirm password:

Confirms the password that is associated with the trusted identity.

Data type: Text

Message layer authentication:

The following options are available for message layer authentication:

Never Specifies that this server cannot accept authentication using any of the mechanisms selected below.

Supported
Specifies that a client communicating with this server can authenticate using any of the mechanisms selected below. However, a method might be invoked without this type of authentication. For example, an anonymous or client certificate might be used instead.

Required
Specifies that clients communicating with this server must specify authentication information using of the mechanisms selected below for any method request.

Allow client to server authentication with::

Specifies client-to-server authentication using Kerberos, LTPA or Basic authentication.

The following options are available for client to server authentication:

Kerberos (KRB5)
Select to specify Kerberos as the authentication mechanism. You must first configure the Kerberos authentication mechanism. Read about Configuring Kerberos as the authentication mechanism using the administrative console for more information.

LTPA Select to configure and enable Lightweight Third-Party Authentication (LTPA) token authentication.

Basic authentication
Basic authentication is Generic Security Services Username Password (GSSUP). This type of authentication typically involves sending a user ID and a password from the client to the server for authentication.

If you select **Basic Authentication** and **LTPA**, and the active authentication mechanism is LTPA, the server goes with a downstream server with a user name, password or LTPA token.

If you select **Basic Authentication** and **KRB5**, and the active authentication mechanism is KRB5, the server goes with a downstream server with a user name, password, Kerberos token or LTPA token.

If you do not select **Basic Authentication**, the server does not go with a downstream server with a user name and password.

Transport:

Specifies whether client processes connect to the server using one of its connected transports.

You can choose to use either Secure Sockets Layer (SSL), TCP/IP or both as the inbound transport that a server supports. If you specify TCP/IP, the server only supports TCP/IP and cannot accept SSL connections. If you specify SSL-supported, this server can support either TCP/IP or SSL connections. If you specify SSL-required, then any server communicating with this one must use SSL.

For server-to-server communication, it is not enough to enable only the Transport layer. You must also enable either the Message layer or the Attribute layer.

TCP/IP
If you select **TCP/IP**, then the server opens a TCP/IP listener port only and all inbound requests do not have SSL protection.

SSL-required
If you select **SSL-required**, then the server opens an SSL listener port only and all inbound requests are received using SSL.

SSL-supported
If you select **SSL-supported**, then the server opens both a TCP/IP and an SSL listener port and most inbound requests are received using SSL.

Provide a fixed port number for the following ports. A zero port number indicates that a dynamic assignment is made at run time.

CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS

Default: SSL-Required
Range: TCP/IP, SSL Required, SSL-Supported

SSL settings:

Specifies a list of predefined SSL settings to choose from for inbound connection.

Data type: String
Default: DefaultSSLSettings
DefaultIIOPSSL
Range: Any SSL settings configured in the SSL Configuration Repertoire

Client certificate authentication:

Specifies whether a client certificate from the configured keystore is used to authenticate to the server when the SSL connection is made between this server and a downstream server, provided that the downstream server supports client certificate authentication.

Typically, client certificate authentication has a higher performance than message layer authentication, but requires some additional setup. These additional steps include verifying that this server has a personal certificate and that the downstream server has the signer certificate of this server.

If you select client certificate authentication, the following options are available:

Never Specifies that this server does not attempt Secure Sockets Layer (SSL) client certificate authentication with downstream servers.

Supported
Specifies that this server can use SSL client certificates to authenticate to downstream servers. However, a method can be invoked without this type of authentication. For example, the server can use anonymous or basic authentication instead.

Required
Specifies that this server must use SSL client certificates to authenticate to downstream servers.

Default: Enabled

Login configuration:

Specifies the type of system login configuration to use for inbound authentication.

You can add custom login modules by clicking **Security > Global security**. From Authentication, click **Java Authentication and Authorization Service > System logins**.

Stateful sessions:

Select this option to enable stateful sessions, which are used mostly for performance improvements.

The first contact between a client and server must fully authenticate. However, all subsequent contacts with valid sessions reuse the security information. The client passes a context ID to the server, and the ID is used to look up the session. The context ID is scoped to the connection, which guarantees uniqueness. Whenever the security session is not valid and the authentication retry is enabled, which is the default, the client-side security interceptor invalidates the client-side session and submits the request again without user awareness. This situation might occur if the session does not exist on the server, for example, the server failed and resumed operation. When this value is disabled, every method invocation must authenticate again.

Enable CSIv2 session cache limit:

Specifies whether to limit the size of the CSIv2 session cache.

When you enable this option, you must set values for the **Maximum cache size** and **Idle session timeout** options. When you do not enable this option, the CSIv2 session cache is not limited.

In previous versions of the application server, you might have set this value as the com.ibm.websphere.security.util.csiv2SessionCacheLimitEnabled custom property. In this product version, it is advisable to set this value using this administrative console panel and not as a custom property.

Default: false

Maximum cache size:

Specify the maximum size of the session cache after which expired sessions are deleted from the cache.

Expired sessions are defined as sessions that are idle longer than the time that is specified in the **Idle session timeout** field. When you specify a value for the **Maximum cache size** field, consider setting its value between 100 and 1000 entries.

Consider specifying a value for this field if your environment uses Kerberos authentication and has a short clock skew for the configured key distribution center (KDC). In this scenario, a short clock skew is defined as less than 20 minutes. Consider increasing the value of this field if the small cache size causes the garbage collection to run so frequently that it impacts the performance of the application server.

In previous versions of the application server, you might have set this value as the com.ibm.websphere.security.util.csiv2SessionCacheMaxSize custom property. In this product version, it is advisable to set this value using this administrative console panel and not as a custom property.

This field only applies if you enable both the **Stateful sessions** and the **Enable CSIv2 session cache limit** options.

Default: By default, a value is not set.
Range: 100 to 1000 entries

Idle session timeout:

This property specifies the time in milliseconds that a CSIv2 session can remain idle before being deleted. The session is deleted if you select the Enable CSIv2 session cache limit option and the value of the **Maximum cache size** field is exceeded.

This timeout value only applies if you enable both the **Stateful sessions** and the **Enable CSIv2 session cache limit** options. Consider decreasing the value for this field if your environment uses Kerberos authentication and has a short clock skew for the configured key distribution center (KDC). In this scenario, a short clock skew is defined as less than 20 minutes. A small clock skew can result in a larger number of rejected CSIv2 sessions. However, with a smaller value for the **Idle session timeout** field, the application server can clean out these rejected sessions more frequently and potentially reduce the resource shortages.

In previous versions of WebSphere Application Server, you might have set this value as the com.ibm.websphere.security.util.csiv2SessionCacheIdleTime custom property. In this product version, it is advisable to set this value using this administrative console panel and not as a custom property. If you previously set it as a custom property, the value was set in milliseconds and converted on this administrative console panel to seconds. On this administrative console panel, you must specify the value in seconds.

Default: By default, a value is not set.
Range: 60 to 86,400 seconds

Custom outbound mapping:

Enables the use of custom Remote Method Invocation (RMI) outbound login modules.

The custom login module maps or completes other functions before the predefined RMI outbound call.

To declare a custom outbound mapping, complete the following steps:
1. Click **Security > Global security**.
2. From Authentication, click **Java Authentication and Authorization Service > System logins > New**.

Trusted authentication realms - outbound:

If the RMI/IIOP communication is across different realms, use this link to add outbound trusted realms.

The credential tokens are only sent to the realms that are trusted. In addition, the receiving server should trust this realm using the inbound trusted realms configuration to validate the LTPA token.

Configuring inbound transports

By using this configuration, you can configure a different transport for inbound security versus outbound security.

Before you begin

Inbound transports refer to the types of listener ports and their attributes that are opened to receive requests for this server. Both Common Secure Interoperability Specification, Version 2 (CSIv2) and Secure Authentication Service (SAS) have the ability to configure the transport.

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

However, the following differences between the two protocols exist:
- CSIv2 is much more flexible than SAS, which requires Secure Sockets Layer (SSL); CSIv2 does not require SSL.
- SAS does not support SSL client certificate authentication, while CSIv2 does.
- CSIv2 can require SSL connections, while SAS only supports SSL connections.
- SAS always has two listener ports open: TCP/IP and SSL.

- CSIv2 can have as few as one listener port and as many as three listener ports. You can open one port for just TCP/IP or when SSL is required. You can open two ports when SSL is supported, and open three ports when SSL and SSL client certificate authentication is supported.

About this task

Complete the following steps to configure the Inbound transport panels in the administrative console:

Procedure

1. Click **Security > Global security**.
2. Under RMI/IIOP security, click **CSIv2 inbound communications**.
3. Under Transport, select **SSL-required**. You can choose to use either Secure Sockets Layer (SSL), TCP/IP or both as the inbound transport that a server supports. If you specify TCP/IP, the server only supports TCP/IP and cannot accept SSL connections. If you specify SSL-supported, this server can support either TCP/IP or SSL connections. If you specify SSL-required, then any server communicating with this one must use SSL.
4. Click **Apply**.
5. Consider fixing the listener ports that you configured.

   You complete this action in a different panel, but think about this action now. Most endpoints are managed at a single location, which is why they do not display in the Inbound transport panels. Managing end points at a single location helps you decrease the number of conflicts in your configuration when you assign the endpoints. The location for SSL end points is at each server. The following port names are defined in the End points panel and are used for Object Request Broker (ORB) security:
   - CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS - CSIv2 Client Authentication SSL Port
   - CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS - CSIv2 SSL Port
   - SAS_SSL_SERVERAUTH_LISTENER_ADDRESS - SAS SSL Port
   - ORB_LISTENER_PORT - TCP/IP Port

   For an application server, click **Servers > Application servers >** *server_name*. Under Communications, click **Ports**. The Ports panel is displayed for the specified server.

   The Object Request Broker (ORB) on WebSphere Application Server uses a listener port for Remote Method Invocation over the Internet Inter-ORB Protocol (RMI/IIOP) communications, and is statically specified using configuration dialogs or during migration. If you are working with a firewall, you must specify a static port for the ORB listener and open that port on the firewall so that communication can pass through the specified port. The endPoint property for setting the ORB listener port is: ORB_LISTENER_ADDRESS.

   a. Click **Servers > Application Servers >** *server_name*. Under Communications, click **Ports > New**.
   b. Select **ORB_LISTENER_ADDRESS** from the **Port name** field in the Configuration panel.
   c. Enter the IP address, the fully qualified Domain Name System (DNS) host name, or the DNS host name by itself in the **Host** field. For example, if the host name is `myhost`, the fully qualified DNS name can be `myhost.myco.com` and the IP address can be `155.123.88.201`.
   d. Enter the port number in the **Port** field. The port number specifies the port for which the service is configured to accept client requests. The port value is used with the host name. Using the previous example, the port number might be 9000.
6. Click **Security > Global security**. Under RMI/IIOP security, click **CSIv2 inbound communications**. Select the SSL settings that are used for inbound requests from CSIv2 clients, and then click **Apply**. Remember that the CSIv2 protocol is used to inter-operate with previous releases. When configuring the keystore and truststore files in the SSL configuration, these files need the right information for inter-operating with previous releases of WebSphere Application Server.

Results

The inbound transport configuration is complete. With this configuration, you can configure a different transport for inbound security versus outbound security. For example, if the application server is the first server that is used by users, the security configuration might be more secure. When requests go to back-end enterprise bean servers, you might lessen the security for performance reasons when you go outbound. With this flexibility you can design the right transport infrastructure to meet your needs.

What to do next

When you finish configuring security, perform the following steps to save, synchronize, and restart the servers:
1. Click **Save** in the administrative console to save any modifications to the configuration.
2. Stop and restart all servers, when synchronized.

Common Secure Interoperability Version 2 transport inbound settings:

Use this page to specify which listener ports to open and which Secure Sockets Layer (SSL) settings to use. These specifications determine which transport a client or upstream server uses to communicate with this server for incoming requests.

To view this administrative console page, complete the following steps:
1. Click **Security > Global security**.
2. Under Authentication, click **RMI/IIOP security > CSIv2 inbound transport**.

Transport:

Specifies whether client processes connect to the server using one of its connected transports.

You can choose to use either Secure Sockets Layer (SSL), TCP/IP or both as the inbound transport that a server supports. If you specify TCP/IP, the server only supports TCP/IP and cannot accept SSL connections. If you specify SSL-supported, this server can support either TCP/IP or SSL connections. If you specify SSL-required, then any server communicating with this one must use SSL.

If you specify SSL-supported or SSL-required, decide which set of SSL configuration settings you want to use for the inbound configuration. This decision determines which key file and trust file are used for inbound connections to this server.

TCP/IP
If you select **TCP/IP**, then the server opens a TCP/IP listener port only and all inbound requests do not have SSL protection.

SSL-required
If you select **SSL-required**, then the server opens an SSL listener port only and all inbound requests are received using SSL.

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

SSL-supported
If you select **SSL-supported**, then the server opens both a TCP/IP and an SSL listener port and most inbound requests are received using SSL.

Provide a fixed port number for the following ports. A zero port number indicates that a dynamic assignment is made at runtime.

CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS

Default: SSL Required
Range: TCP/IP, SSL Required, SSL-Supported

SSL settings:

Specifies a list of predefined SSL settings to choose from for inbound connections.

These settings are configured at the SSL Repertoire panel. To access the SSL Repertoire panel, complete the following steps:

1. Clicking **Security > SSL certificate and key management**.
2. Under configuration settings, click **Manage endpoint security configurations and trust zones**.
3. Expand Inbound and click *inbound_configuration*.
4. Under Related items, click **SSL configurations**.

Data type: String
Default: DefaultSSLSettings
DefaultIIOPSSL
Range: Any SSL settings configured in the SSL Configuration Repertoire

Centrally managed:

Specifies that the selection of an SSL configuration is based upon the outbound topology view for the Java Naming and Directory Interface (JNDI) platform.

Centrally managed configurations support one location to maintain SSL configurations rather than spreading them across the configuration documents.

Default: Enabled

Use specific SSL alias:

Specifies the SSL configuration alias to use for LDAP outbound SSL communications.

This option overrides the centrally managed configuration for the JNDI platform.

z/OS SSL settings:

Specifies a list of predefined Secure Sockets Layer (SSL) settings for inbound connections. Configure these settings on the SSL panel by clicking Secure communications on the administrative console.

Secure Authentication Service inbound transport settings:

Use this page to specify transport settings for connections that are accepted by this server using the Secure Authentication Service (SAS) authentication protocol. The SAS protocol is used to communicate securely to enterprise beans with previous releases of the application server.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.

2. Under Authentication, expand RMI/IIOP security and click **SAS inbound transport**.

Attention: The panel associated with this article displays only when you have a Version 6.1 server in your environment. SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

SSL Settings:

Specifies a list of predefined SSL settings to choose from for inbound connections.

These settings are configured on the Secure Sockets Layer (SSL) configuration panel. To access the SSL configuration panel, complete the following steps:

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations and trust zones**.
2. Expand Inbound > *configuration_name*.
3. Under Related Items, click **SSL configurations**.

Data type:	String
Default:	DefaultSSLSettings

Configuring outbound transports

By using this configuration, you can configure a different transport for inbound security versus outbound security.

Before you begin

Outbound transports refers to the transport that is used to connect to a downstream server. When you configure the outbound transport, consider the transports that the downstream servers support. If you are considering Secure Sockets Layer (SSL), also consider including the signers of the downstream servers in this server truststore file for the handshake to succeed.

When you select an SSL configuration, that configuration points to keystore and truststore files that contain the necessary signers.

If you configured client certificate authentication for this server by completing the following steps, then the downstream servers contain the signer certificate belonging to the server personal certificate:

1. Click **Security > Global security**.
2. Under RMI/IIOP security, click **CSIv2 outbound communications**.

About this task

Complete the following steps to configure the outbound transport panels.

Procedure

1. Select the type of transport and the SSL settings by clicking **Security > Global security**. Under RMI/IIOP security, click **CSIv2 outbound communications**. By selecting the type of transport, you choose the transport to use when connecting to downstream servers. The downstream servers support the transport that you choose. If you choose **SSL-Supported**, the transport that is used is negotiated during the connection. If both the client and server support SSL, always select the **SSL-Supported** option unless the request is considered a special request that does not require SSL, such as if an object request broker (ORB) is a request.
2. Select the **SSL required** option if you want to use Secure Sockets Layer communications with the outbound transport.

If you select the **SSL required** option or the **SSL supported** option, you can select either the **Centrally managed** or **Use specific SSL alias** option.

Centrally managed

Enables you to specify an SSL configuration for particular scope such as the cell, node, server, or cluster in one location. To use the **Centrally managed** option, you must specify the SSL configuration for the particular set of endpoints. The Manage endpoint security configurations and trust zones panel displays all of the inbound and outbound endpoints that use the SSL protocol. If you expand the Inbound or Outbound section of the panel and click the name of a node, you can specify an SSL configuration that is used for every endpoint on that node. For an outbound transport, you can override the inherited SSL configuration by specifying an SSL configuration for a particular endpoint. To specify an SSL configuration for an outbound transport, click **Security > SSL certificate and key management > Manage endpoint security configurations and trust zones** and expand **Outbound**.

Use specific SSL alias

Select the **Use specific SSL alias** option if you intend to select one of the SSL configurations in the menu below the option.

The default is *DefaultSSLSettings*. To modify or create a new SSL configuration, complete the steps described in "Creating a Secure Sockets Layer configuration" on page 1741.

3. Click **Apply**.

Results

The outbound transport configuration is complete. With this configuration, you can configure a different transport for inbound security versus outbound security. For example, if the application server is the first server used by end users, the security configuration might be more secure. When requests go to back-end enterprise beans servers, you might consider less security for performance reasons when you go outbound. With this flexibility you can design a transport infrastructure that meets your needs.

What to do next

When you finish configuring security, perform the following steps to save, synchronize, and restart the servers.

- Click **Save** in the administrative console to save any modifications to the configuration.
- Stop and restart all servers, after synchronization.

Common Secure Interoperability Version 2 outbound transport settings:

Use this page to specify which transports and Secure Sockets Layer (SSL) settings this server uses when communicating with downstream servers for outbound requests.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under Authentication, click **RMI/IIOP security > CSIv2 outbound transport**.

You also can view this administrative console by completing the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **Server security**.
3. Under Additional properties, click **CSIv2 outbound transport**.

Transport:

Specifies whether the client processes connect to the server using one of the server-connected transports.

You can choose to use either SSL, TCP/IP, or Both as the outbound transport that a server supports. If you specify `TCP/IP`, the server supports only TCP/IP and cannot initiate SSL connections with downstream servers. If you specify `SSL-supported`, this server can initiate either TCP/IP or SSL connections. If you specify `SSL-required`, this server must use SSL to initiate connections to downstream servers. When you do specify SSL, decide which set of SSL configuration settings you want to use for the outbound configuration.

This decision determines which keyfile and trustfile to use for outbound connections to downstream servers.

Consider the following options:

TCP/IP
If you select this option, the server opens TCP/IP connections with downstream servers only.

SSL-required
If you select this option, the server opens SSL connections with downstream servers.

SSL-supported
If you select this option, the server opens SSL connections with any downstream server that supports them and opens TCP/IP connections with any downstream servers that do not support SSL.

Default: SSL-supported
Range: TCP/IP, SSL-required, SSL-supported

SSL settings:

Specifies a list of predefined SSL settings for outbound connections. These settings are configured at the SSL Configuration Repertoires panel.

To access the panel, complete the following steps:

1. Click **Security > SSL certificate and key management**.
2. Under Configuration settings, click **Manage endpoint security configurations and trust zones**.
3. Expand Outbound > *outbound_configuration_name*.
4. Under Related items, click **SSL configurations**.

Data type: String
Range: Any SSL settings that are configured in the SSL Configuration Repertoires panel

Note: This field is available only if a Version 6.1 server exists in your environment.

SSL enabled:

Specifies whether secure socket communication is enabled to the server.

Centrally managed:

Specifies that the selection of an SSL configuration is based upon the outbound topology view for the Java Naming and Directory Interface (JNDI) platform.

Centrally managed configurations support one location to maintain SSL configurations rather than spreading them across the configuration documents.

Default: Enabled

Use specific SSL alias:

Specifies the SSL configuration alias that you want to use for outbound SSL communications.

This option overrides the centrally managed configuration for the JNDI (LDAP) protocol.

Secure Authentication Service outbound transport settings:

Use this page to specify transport settings for connections that are accepted by this server using the Secure Authentication Service (SAS) authentication protocol.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Under Authentication, expand RMI/IIOP security and click **SAS outbound transport**.

Attention: The panel associated with this article displays only when you have a Version 6.1 server in your environment.

SSL settings:

Specifies a list of predefined Secure Sockets Layer (SSL) settings to choose from for outbound connections.

These settings are configured on the SSL configuration panel. To access the SSL configuration panel, complete the following steps:

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations and trust zones**.
2. Expand Outbound > *configuration_name*.
3. Under Related Items, click **SSL configurations**.

Data type:	String
Default:	DefaultSSLSettings

Configuring inbound messages

You can use the administrative console to configure inbound messages for CSIv2.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under Authentication, expand **RMI/HOP security**.
3. Click **CSIv2 inbound communication**.
4. Optional: Click **Propagate security attributes** or **Use identity assertion**. The **Propagate security attributes** option enables support for security attribute propagation during login requests. When you select this option, the application server retains additional information about the login request, such as the authentication strength used, and retains the identity and location of the request originator.

   The **Use identity assertion** option specifies that identity assertion is a way to assert identities from one server to another during a downstream Enterprise JavaBeans (EJB) invocation.
5. Under CSIv2 Message layer authentication, select **Supported**, **Never** or **Required**.

   Never Specifies that this server cannot accept an authentication mechanism that you select under **Allow client to server authentication with:**.

   Supported
   Specifies that clients communicating with this server can specify an authentication mechanism

that you select under **Allow client to server authentication with:**. However, a method might be invoked without this type of authentication. For example, an anonymous or client certificate might be used instead.

Required
Specifies that clients communicating with this server must specify an authentication mechanism that you select under **Allow client to server authentication with:**.

6. Under **Allow client to server authentication with:**, select **Kerberos**, **LTPA** and or **Basic authentication**. You can optionally select:

   Kerberos
   Select to enable authentication using the Kerberos token.

   LTPA Select to enable authentication using the Lightweight Third-Party Authentication (LTPA) token.

   Basic authentication
   This type of authentication typically involves sending a user ID and a password from the client to the server for authentication. This is also know as Generic Security Services Username Password (GSSUP).

   This authentication also involves delegating a credential token from an already authenticated credential, provided the credential type is forwardable; for example, LTPA.

   If you select **supported** under **CSIv2 Message layer authentication**, and check **KRB5** and **LTPA** under **Allow client to server authentication with:**, then the server does not accept the user name and password.

7. Click **OK.**

Results

You have now configured messages for CSIv2 inbound.

Configuring outbound messages

You can use the administrative console to configure outbound messages for CSIv2.

Procedure

1. In the administrative console, click **Security > Global security**.
2. Under Authentication, expand **RMI/HOP security**.
3. Click **CSIv2 outbound communication**.
4. Optional: Click **Propagate security attributes** or **Use identity assertion**. The **Propagate security attributes** option enables support for security attribute propagation during login requests. When you select this option, the application server retains additional information about the login request, such as the authentication strength used, and retains the identity and location of the request originator.

   The **Use identity assertion** option specifies that identity assertion is a way to assert identities from one server to another during a downstream Enterprise JavaBeans (EJB) invocation.

   The **Use server trusted identity** option specifies the server identity that the application server uses to establish trust with the target server.

   The **Specify an alternative trusted identity** option enables you to specify an alternative user as the trusted identity that is sent to the target servers instead of sending the server identity. If you select this option you must provide the name of the trusted identity and the password that is associated with the trusted identity.

   Note: You must select Basic Authentication under the Message Layer authentication section to send an alternative trusted identity. If you do not select Basic Authentication, then choose the Server Identity instead.

5. Under CSIv2 Message layer authentication, select **Supported**, **Never** or **Required**.

Never Specifies that this server cannot accept an authentication mechanism that you select under **Allow client to server authentication with:**.

Supported
Specifies that clients communicating with this server can specify an authentication mechanism that you select under **Allow client to server authentication with:**. However, a method might be invoked without this type of authentication. For example, an anonymous or client certificate might be used instead.

Required
Specifies that clients communicating with this server must specify an authentication mechanism that you select under **Allow client to server authentication with:**.

6. Under **Allow client to server authentication with:**, select **Kerberos**, **LTPA** and or **Basic authentication**. You can optionally select:.

   Kerberos
   Select to enable authentication using the Kerberos token.

   LTPA Select to enable authentication using the Lightweight Third-Party Authentication (LTPA) token.

   Basic authentication
   This type of authentication typically involves sending a user ID and a password from the client to the server for authentication. This is also know as Generic Security Services Username Password (GSSUP).

   This authentication also involves delegating a credential token from an already authenticated credential, provided the credential type is forwardable; for example, LTPA.

   If you select **supported** under **CSIv2 Message layer authentication**, and check **KRB5** and **LTPA** under **Allow client to server authentication with:**, then the server does not accept the user name and password.

7. Optional: Select **Custom outbound mapping**. This option enables the use of custom Remote Method Invocation (RMI) outbound login modules.

Results

You have now configured messages for CSIv2 outbound.

Common Secure Interoperability Version 2 and Security Authentication Service (SAS) client configuration

A secure Java client requires configuration properties to determine how to perform security with a server.

These configuration properties are typically put into a properties file somewhere on the client system and referenced by specifying the following system property on the command line of the Java client. For example, this property accepts any valid web address.

```
-Dcom.ibm.CORBA.ConfigURL=file:profile_root/properties/sas.client.props
```

When you use thin or thick clients, `com.ibm.CORBA.ConfigURL` is automatically set to the following file:

profile_root`/properties/sas.client.props`

When this file is processed by the Object Request Broker (ORB), security can be enabled between the Java client and the target server.

If any syntax problems exist with the ConfigURL property and the `sas.client.props` file is not found, the Java client proceeds to connect insecurely. Errors display indicating the failure to read the ConfigURL property. Typically the problem is related to having two slashes after `file`, which is not valid.

Use the following properties to configure the Secure Authentication Service (SAS) and CSIv2 authentication protocols:
• "Security Authentication Service authentication protocol client settings" on page 1588

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Authentication protocol settings for a client configuration:

You can use settings in the `sas.client.props` file to configure Security Authentication Service (SAS) and Common Secure Interoperability Version 2 (CSIv2) clients.

Use the following settings in the `sas.client.props` file to configure SAS and CSIv2 clients. By default, the `sas.client.props` file is located in the *profile_root*`/properties` directory of your WebSphere Application Server - Express installation.

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Note: The `sas.client.props` file for WebSphere Application Server Version 8.0 contains some new properties that support BasicAuth and Kerberos, such as:

```
com.ibm.IPC.authenticationTarget=BasicAuth
com.ibm.IPC.loginUserid=
com.ibm.IPC.loginPassword=
com.ibm.IPC.loginSource=prompt
com.ibm.IPC.krb5Service=WAS
com.ibm.IPC.krb5CcacheFile=
com.ibm.IPC.krb5ConfigFile=
```

com.ibm.CORBA.securityEnabled:

Use to determine if security is enabled for the client process.

Table 110. com.ibm.CORBA.securityEnabled. This table describes the com.ibm.CORBA.securityEnabled setting.

Setting	Value
Data Type	Boolean
Default	True
Valid values	True or false

com.ibm.CSI.protocol:

Use to determine which authentication protocols are active.

The client can configure protocols of `ibm`, `csiv2` or `both` as active. The only possible values for an authentication protocol are `ibm`, `csiv2` and `both`. Do not use `sas` for the value of an authentication protocol. This restriction applies to both client and server configurations. The following list provides information about using each of these protocol options:

ibm Use this authentication protocol option when you are communicating with WebSphere Application Server Version 4.x or earlier servers.

csiv2 Use this authentication protocol option when you are communicating with WebSphere Application Server Version 5 or later servers because the SAS interceptors are not loaded and running for each method request.

both Use this authentication protocol option for interoperability between WebSphere Application Server

Version 4.x or earlier servers and WebSphere Application Server Version 5 or later servers. Typically, specifying `both` provides greater interoperability with other servers.

Table 111. com.ibm.CSI.protocol. This table describes the com.ibm.CSI.protocol setting.

Setting	Value
Data type	String
Default	Both
Valid values	ibm, csiv2, both

com.ibm.CORBA.authenticationTarget:

Use to determine the type of authentication mechanism for sending security information from the client to the server.

If basic authentication is specified, the user ID and password are sent to the server. Using the Secure Sockets Layer (SSL) transport with this type of authentication is recommended; otherwise, the password is not encrypted. The target server must support the specified authentication target.

Table 112. com.ibm.CORBA.authenticationTarget. This table describes the com.ibm.CORBA.authenticationTarget setting.

Setting	Value
Data type	String
Default	BasicAuth
Valid values	BasicAuth, KRB5

com.ibm.CORBA.validateBasicAuth:

Use to determine if the user ID and password get validated immediately after the login data is entered when the authenticationTarget property is set to `BasicAuth`.

In previous releases, BasicAuth logins validated only with the initial method request. During the first request, the user ID and password are sent to the server. This request is the first time that the client can notice an error, if the user ID or password is incorrect. The validateBasicAuth method is specified and the validation of the user ID and password occurs immediately to the security server.

For performance reasons, you might want to disable this property if you do not want to verify the user ID and password immediately. If the client program can wait, it is better to have the initial method request flow to the user ID and password. However, program logic might not be this simple because of error handling considerations.

Table 113. com.ibm.CORBA.validateBasicAuth. This table describes the com.ibm.CORBA.validateBasicAuth setting.

Setting	Value
Data type	Boolean
Default	True
Valid values	True, False

com.ibm.CORBA.authenticationRetryEnabled:

Use to specify that a failed login attempt is retried. This property determines if a retry occurs for other errors, such as stateful sessions that are not found on a server or validation failures at the server because of an expiring credential.

The minor code in the exception that is returned to a client determines which errors are retried. The number of retry attempts is dependent upon the com.ibm.CORBA.authenticationRetryCount property.

Table 114. com.ibm.CORBA.authenticationRetryEnabled. This table describes the com.ibm.CORBA.authenticationRetryEnabled setting.

Setting	Value
Data type	Boolean
Default	True
Valid values	True, False

com.ibm.CORBA.authenticationRetryCount:

Use to specify the number of retries that occur until either a successful authentication occurs or the maximum retry value is reached.

When the maximum retry value is reached, the authentication exception is returned to the client.

Table 115. com.ibm.CORBA.authenticationRetryCount. This table describes the com.ibm.CORBA.authenticationRetryCount setting.

Setting	Value
Data type	Integer
Default	3
Range	1-10

com.ibm.CORBA.loginSource:

Use to specify how the request interceptor attempts to log in if it does not find an invocation credential already set.

This property is valid only if message layer authentication occurs. If only transport layer authentication occurs, this property is ignored. When specifying properties, the following two additional properties must be defined:

- com.ibm.CORBA.loginUserid
- com.ibm.CORBA.loginPassword

When performing a programmatic login, it is not necessary to specify `none` as the login source. The request fails if a credential is set as the invocation credential during a method request.

Table 116. com.ibm.CORBA.loginSource. This table describes the com.ibm.CORBA.loginSource setting.

Setting	Value
Data type	String
Default	Prompt
Valid values	Prompt, key file, stdin, none, properties

com.ibm.CORBA.loginUserid:

Use to specify the user ID when a properties login is configured and message layer authentication occurs.

This property is valid only when `com.ibm.CORBA.loginSource=properties`. Also set the com.ibm.CORBA.loginPassword property.

Table 117. com.ibm.CORBA.loginUserid. This table describes the com.ibm.CORBA.loginUserid setting.

Setting	Value
Data type	String
Range	Any string that is appropriate for a user ID in the configured user registry of the server.

com.ibm.CORBA.loginPassword:

Use to specify the password when a properties login is configured and message layer authentication occurs.

This property is valid only when `com.ibm.CORBA.loginSource=properties`. Also set the com.ibm.CORBA.loginUserid property.

Table 118. com.ibm.CORBA.loginPassword. This table describes the com.ibm.CORBA.loginPassword setting.

Setting	Value
Data type	String
Range	Any string that is appropriate for a password in the configured user registry of the server.

com.ibm.CORBA.keyFileName:

Use to specify the key file that is used to log in.

A key file is a file that contains a list of realm, user ID, and password combinations that a client uses to log into multiple realms. The realm that is used is the one found in the interoperable object reference (IOR) for the current method request. The value of this property is used when the `com.ibm.CORBA.loginSource=key file` is used.

Table 119. com.ibm.CORBA.keyFileName. This table describes the com.ibm.CORBA.keyFileName setting.

Setting	Value
Data type	String
Default	C;/WebSphere/AppServer/properties/wsserver.key
Range	Any fully qualified path and file name of a WebSphere Application Server key file.

com.ibm.CORBA.loginTimeout:

Use to specify the length of time that the login prompt stays available before it is considered a failed login.

Table 120. com.ibm.CORBA.loginTimeout. This table describes the com.ibm.CORBA.loginTimeout setting.

Setting	Value
Data type	Integer
Units	Seconds
Default	300 (5 minute intervals)
Range	0 - 600 (10 minute intervals)

com.ibm.CORBA.securityEnabled:

Use to determine if security is enabled for the client process.

Table 121. com.ibm.CORBA.securityEnabled. This table describes the com.ibm.CORBA.securityEnabled setting.

Setting	Value
Data type	Boolean
Default	True
Range	True, False

Security Authentication Service authentication protocol client settings:

In addition to those properties which are valid for both Security Authentication Service (SAS) and Common Secure Interoperability Version 2 (CSIv2), this article documents properties which are valid only for the SAS authentication protocol.

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

com.ibm.CORBA.standardPerformQOPModels:

Specifies the strength of the ciphers when making a Secure Sockets Layer (SSL) connection.

Data type:	String
Default:	High
Range	Low, Medium, High

Example 1: Configuring basic authentication and identity assertion

This example presents a pure Java client, C, that accesses a secure enterprise bean on server, S1, through user bob. The following steps take you through the configuration of C, S1, and S2.

About this task

The enterprise bean code on S1 accesses another enterprise bean on server, S2. This configuration uses identity assertion to propagate the identity of bob to the downstream server, S2. S2 trusts that bob already is authenticated by S1 because it trusts S1. To gain this trust, the identity of S1 also flows to S2 simultaneously and S2 validates the identity by checking the trustedPrincipalList list to verify that it is a valid server principal. S2 also authenticates S1.

Procedure

1. Configure the client C for message layer authentication with a Secure Sockets Layer (SSL) transport.
   a. Point the client to the `sas.client.props` file.

      Use the `com.ibm.CORBA.ConfigURL=file:/`*`profile_root`*` /properties/sas.client.props` property. The *profile_root* variable is the specific profile that you are working with. All further configuration involves setting properties within this file.
   b. Enable SSL.

      In this case, SSL is supported but not required:
      `com.ibm.CSI.performTransportAssocSSLTLSSupported=true,`
      `com.ibm.CSI.performTransportAssocSSLTLSRequired=false`
   c. Enable client authentication at the message layer.

      In this case, client authentication is supported but not required:
      `com.ibm.CSI.performClientAuthenticationRequired=false,`
      `com.ibm.CSI.performClientAuthenticationSupported=true`
   d. Use all of the remaining defaults in the `sas.client.props` file.
2. Configure the server, S1.

   In the administrative console, server S1 is configured for incoming requests to support message-layer client authentication and incoming connections to support SSL without client certificate authentication. Server S1 is configured for outgoing requests to support identity assertion.
   a. Configure S1 for incoming connections.
      1) Disable identity assertion.
      2) Enable user ID and password authentication.
      3) Enable SSL.

4) Disable SSL client certificate authentication.

b. Configure S1 for outgoing connections.

1) Enable identity assertion.

2) Disable user ID and password authentication.

3) Enable SSL.

4) Disable SSL client certificate authentication.

3. Configure the server, S2.

In the administrative console, server S2 is configured for incoming requests to support identity assertion and to accept SSL connections. Complete the following steps to configure incoming connections. Configuration for outgoing requests and connections are not relevant for this example.

a. Enable identity assertion.

b. Disable user ID and password authentication.

c. Enable SSL.

d. Disable SSL client authentication.

Example 2: Configuring basic authentication, identity assertion, and client certificates

This example is the same as example 1, except for the interaction from client C2 to server S2. Therefore, the configuration of example 1 still is valid, but you have to modify server S2 slightly and add a configuration for client C2. The configuration is not modified for C1 or S1.

About this task

Procedure

1. Configure client C2 for transport layer authentication (Secure Sockets Layer (SSL) client certificates).

a. Point the client to the `sas.client.props` file.

Use the `com.ibm.CORBA.ConfigURL=file:/profile_root /properties/sas.client.props` property. The *profile_root* variable is the specific profile that you are working with. All further configuration involves setting properties within this file.

b. Enable SSL.

In this case, SSL is supported but not required:

```
com.ibm.CSI.performTransportAssocSSLTLSSupported=true,
com.ibm.CSI.performTransportAssocSSLTLSRequired=false
```

c. Disable client authentication at the message layer.

```
com.ibm.CSI.performClientAuthenticationRequired=false,
com.ibm.CSI.performClientAuthenticationSupported=false
```

d. Enable client authentication at the transport layer where it is supported, but not required.

```
com.ibm.CSI.performTLClientAuthenticationRequired=false,
com.ibm.CSI.performTLClientAuthenticationSupported=true
```

2. Configure the server, S2.

In the administrative console, server S2 is configured for incoming requests to SSL client authentication and identity assertion. Configuration for outgoing requests is not relevant for this example.

You can mix and match these configuration options. However, a precedence exists as to which authentication features become the identity in the received credential:

a. Identity assertion

b. Message-layer client authentication (basic authentication or token)
c. Transport-layer client authentication (SSL certificates)

a. Enable identity assertion.
b. Disable user ID and password authentication.
c. Enable SSL.
d. Enable SSL client authentication.

Example 3: Configuring client certificate authentication and RunAs system

This example presents a pure Java client, C, accessing a secure enterprise bean on S1.

About this task

C authenticates to S1 using Secure Sockets Layer (SSL) client certificates. S1 maps the common name of the distinguished name (DN) in the certificate to a user in the local registry. The user in this case is bob. The enterprise bean code on S1 accesses another enterprise bean on S2. Because the RunAs mode is `system`, the invocation credential is set as server1 for any outbound requests.

Procedure

1. Configure client C for transport layer authentication (SSL client certificates).
   a. Point the client to the `sas.client.props` file.

      Use the `com.ibm.CORBA.ConfigURL=file:/`*profile_root* `/properties/sas.client.props` property. The *profile_root* variable is the specific profile that you are working with. All further configuration involves setting properties within this file.
   b. Enable SSL.

      In this case, SSL is supported but not required:
      `com.ibm.CSI.performTransportAssocSSLTLSSupported=true,`
      `com.ibm.CSI.performTransportAssocSSLTLSRequired=false`
   c. Disable client authentication at the message layer.
      `com.ibm.CSI.performClientAuthenticationRequired=false,`
      `com.ibm.CSI.performClientAuthenticationSupported=false`
   d. Enable client authentication at the transport layer. It is supported, but not required.
      `com.ibm.CSI.performTLClientAuthenticationRequired=false,`
      `com.ibm.CSI.performTLClientAuthenticationSupported=true`
2. Configure the S1 server. In the administrative console, S1 is configured for incoming connections to support SSL with client certificate authentication. The S1 server is configured for outgoing requests to support message layer client authentication.
   a. Configure S1 for incoming connections.
      1) Disable identity assertion.
      2) Disable user ID and password authentication.
      3) Enable SSL.
      4) Enable SSL client certificate authentication.
   b. Configure S1 for outgoing connections.
      1) Disable identity assertion.
      2) Disable user ID and password authentication.
      3) Enable SSL.
      4) Enable SSL client certificate authentication.
3. Configure the S2 server.

In the administrative console, the S2 server is configured for incoming requests to support message layer authentication over SSL. Configuration for outgoing requests is not relevant for this scenario.

a. Disable identity assertion.

b. Enable user ID and password authentication.

c. Enable SSL.

d. Disable SSL client authentication.

Example 4: Configuring TCP/IP transport using a virtual private network

This scenario illustrates the ability to choose TCP/IP as the transport when it is appropriate. In some cases, when two servers are on the same virtual private network (VPN), it can be appropriate to select TCP/IP as the transport for performance reasons because the VPN already encrypts the message.

About this task

Procedure

1. Configure client C for message layer authentication with an Secure Sockets Layer (SSL) transport.
   a. Point the client to the `sas.client.props` file.

      Use the `com.ibm.CORBA.ConfigURL=file:/`*profile_root*`/properties/sas.client.props` property. The *profile_root* variable is to the specific profile you are working with. All further configuration involves setting properties within this file.

   b. Enable SSL.

      In this case, SSL is supported but not required.`com.ibm.CSI.performTransportAssocSSLTLSSupported=true`, `com.ibm.CSI.performTransportAssocSSLTLSRequired=false`

   c. Enable client authentication at the message layer. In this case, client authentication is supported but not required. `com.ibm.CSI.performClientAuthenticationRequired=false`, `com.ibm.CSI.performClientAuthenticationSupported=true`

   d. Use the remaining defaults in the `sas.client.props` file.
2. Configure the S1 server. In the administrative console, the S1 server is configured for incoming requests to support message-layer client authentication and incoming connections to support SSL without client certificate authentication. The S1 server is configured for outgoing requests to support identity assertion.

   It is possible to enable SSL for inbound connections and disable SSL for outbound connections. The same is true in reverse.
   a. Configure S1 for incoming connections.
      1) Disable identity assertion.
      2) Enable user ID and password authentication.
      3) Enable SSL.
      4) Disable SSL client certificate authentication.
   b. Configure S1 for outgoing connections.
      1) Disable identity assertion.
      2) Enable user ID and password authentication.
      3) Disable SSL.
3. Configure the S2 server.

   In the administrative console, the S2 server is configured for incoming requests to support identity assertion and to accept SSL connections. Configuration for outgoing requests and connections are not relevant for this scenario.
   a. Disable identity assertion.

b. Enable user ID and password authentication.
c. Disable SSL.

Authentication protocol for EJB security

WebSphere Application Server Version 8.0 servers support the CSIv2 authentication protocol only. SAS is only supported between Version 6.0.x and earlier version servers that have been federated in a Version 8.0 cell. The option to select between SAS, CSIv2, or both is only available in the administration console when a Version 6.0.*x* or earlier release has been federated in a Version 8.0 cell.

SAS is the authentication protocol used by all previous releases of WebSphere Application Server and is maintained for backwards compatibility. The Object Management Group (OMG) has defined the authentication protocol called CSIv2 so that vendors can interoperate securely. CSIv2 is implemented in WebSphere Application Server with more features than SAS and is considered the strategic protocol.

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Invoking Enterprise Java Beans (EJB) methods in a secure WebSphere Application Server environment requires an authentication protocol to determine the level of security and the type of authentication that occur between any given client and server for each request. It is the job of the authentication protocol during a method invocation to merge the server authentication requirements that are determined by the object Interoperable Object Reference (IOR) with the client authentication requirements that are determined by the client configuration and come up with an authentication policy specific to that client and server pair.

The authentication policy makes the following decisions, among others, which are all based on the client and server configurations:
- What kind of connection can you make to this server--Secure Sockets Layer (SSL) or TCP/IP?
- If SSL is chosen, how strong is the encryption of the data?
- If SSL is chosen, do you authenticate the client using client certificates?
- Do you authenticate the client with a user ID and password? Does an existing credential exist?
- Do you assert the client identity to downstream servers?
- Given the configuration of the client and server, can a secure request proceed?

You can configure both protocols (SAS and CSIv2) to work simultaneously. If a server supports both protocols, it exports an IOR containing tagged components describing the configuration for SAS and CSIv2. If a client supports both protocols, it reads tagged components for both CSIv2 and SAS. If the client supports both and the server supports both, CSIv2 is used. However, if the server supports SAS (for example, it is a previous WebSphere Application Server release) and the client supports both, the client chooses SAS for this request because the SAS protocol is what both have in common.

Choose a protocol by specifying the `com.ibm.CSI.protocol` property on the client side and configuring through the administrative console on the server side. More details are included in the SAS and CSIv2 properties articles.

Common Secure Interoperability Specification, Version 2

The Common Secure Interoperability Specification, Version 2 (CSIv2) defines the Security Attribute Service (SAS) that enables interoperable authentication, delegation, and privileges. The CSIv2 SAS and SAS protocols are entirely different. The CSIv2 SAS is a subcomponent of CSIv2 that supports SSL and interoperability with the EJB Specification, Version 2.1.

Security Attribute Service

The Common Secure Interoperability Specification, Version 2 Security Attribute Service (CSIv2 SAS) protocol is designed to exchange its protocol elements in the service context of a General Inter-ORB Protocol (GIOP) request and reply messages that are communicated over a connection-based transport. The protocol is intended for use in environments where transport layer security, such as that available through Secure Sockets Layer (SSL) and Transport Layer Security (TLS), is used to provide message protection (that is, integrity and or confidentiality) and server-to-client authentication. The protocol provides client authentication, delegation, and privilege functionality that might be applied to overcome corresponding deficiencies in an underlying transport. The CSIv2 SAS protocol facilitates interoperability by serving as the higher-level protocol under which secure transports can be unified.

Connection and request interceptors

The authentication protocols that are used by WebSphere Application Server are add-on Interoperable Inter-ORB Protocol (IIOP) services. IIOP is a request-and-reply communications protocol that is used to send messages between two Object Request Brokers (ORBs). For each request made by a client ORB to a server ORB, an associated reply is made by the server ORB back to the client ORB. Prior to any request flowing, a connection between the client ORB and the server ORB must be established over the TCP/IP transport (SSL is a secure version of TCP/IP). The client ORB invokes the authentication protocol client connection interceptor, which is used to read the tagged components in the IOR of the object that is located on the server. As mentioned previously, the authentication policy is established here for the request. Given the authentication policy (a coalescing of the server configuration with the client configuration), the strength of the connection is returned to the ORB. The ORB makes the appropriate connection, usually over SSL.

After the connection is established, the client ORB invokes the authentication protocol client request interceptor, which is used to send security information other than what is established by the transport. The security information includes the user ID and password token that are authenticated by the server, an authentication mechanism-specific token that is validated by the server, or an identity assertion token. Identity assertion is a way for one server to trust another server without the need to re-authenticate or re-validate the originating client. However, some work is required for the server to trust the upstream server. This additional security information is sent with the message in a *service context*. A service context has a registered identifier so that the server ORB can identify which protocol is sending the information.

The fact that a service context contains a unique identity is another way for WebSphere Application Server to support both SAS and CSIv2 simultaneously because both protocols have different service context IDs. After the client request interceptor finishes adding the service context to the message, the message is sent to the server ORB.

When the message is received by the server ORB, the ORB invokes the authentication protocol server request interceptor. This interceptor looks for the service context ID known by the protocol. When both SAS and CSIv2 are supported by a server, two different server request interceptors are invoked and both interceptors look for different service context IDs.

However, only one finds a service context for any given request. When the server request interceptor finds a service context, it reads the information in the service context. A method is invoked to the security server to authenticate or validate client identity. The security server either rejects the information or returns a credential. A credential contains additional information about the client that is retrieved from the user registry so that authorization can make the appropriate decision. Authorization is the process of determining if the user can invoke the request based on the roles that are applied to the method and the roles given to the user.

If a service context is not found by the CSIv2 server request interceptor, the interceptor process looks at the transport connection to see if a client certificate chain is sent. This process is done when SSL client authentication is configured between the client and server.

If a client certificate chain is found, the distinguished name (DN) is extracted from the certificate and is used to map to an identity in the user registry. If the user registry is Lightweight Directory Access Protocol (LDAP), the search filters defined in the LDAP registry configuration determine how the certificate maps to an entry in the registry. If the user registry is local OS, the first attribute of the distinguished name (DN) maps to the user ID of the registry. This attribute is typically the common name.

If the certificate does not map, no credential is created and the request is rejected. When valid security information is not presented, the method request is rejected and a NO_PERMISSION exception is sent back with the reply. However, when no security information is presented, an unauthenticated credential is created for the request and the authorization engine determines if the method gets invoked. For an unauthenticated credential to invoke an Enterprise JavaBeans (EJB) method, either no security roles are defined for the method or a special Everyone role is defined for the method.

When the method invocation is completed in the EJB container, the server request interceptor is invoked again to complete server authentication and a new reply service context is created to inform the client request interceptor of the outcome. This process is typically for making the request *stateful*. When a stateful request is made, only the first request between a client and server requires that security information is sent. All subsequent method requests need to send a unique context ID only so that the server can look up the credential that is stored in a session table. The context ID is unique within the connection between a client and server.

Finally, the method request cycle is completed by the client request interceptor receiving a reply from the server with a reply service context providing information so that the client-side stateful context ID can be confirmed and reused.

Specifying a stateful client is done through the property com.ibm.CSI.performStateful (true/false). Specifying a stateful server is done through the administrative console configuration.

. *Authentication protocol flow*

Authentication policy for each request

The authentication policy of a given request determines the security protection between a client and a server. A client or server authentication protocol configuration can describe required features, supported features, and non-supported features. When a client requires a feature, it can talk only to servers that either require or support that feature. When a server requires a feature, it can talk only to clients that either require or support that feature. When a client supports a feature, it can talk to a server that supports or requires that feature, but can also talk to servers that do not support the feature. When a server supports a feature, it can talk to a client that supports or requires the feature, but can also talk to clients that do not support the feature or chose not to support the feature.

For example, for a client to support client certificate authentication, some setup is required to either generate a self-signed certificate or to get one from a certificate authority (CA). Some clients might not need to complete these actions, therefore, you can configure this feature as not supported. By making this decision, the client cannot communicate with a secure server that requires client certificate authentication. Instead, this client can choose to use the user ID and password as the method of authenticating itself to the server.

Typically, supporting a feature is the most common way of configuring features. It is also the most successful during runtime because it is more forgiving than requiring a feature. Knowing how secure servers are configured in your domain, you can choose the right combination for the client to ensure successful method invocations and still get the most security. If you know that all of your servers support both client certificate and user ID and password authentication for the client, you might want to require one and not support the other. If both the user ID and password and the client certificate are supported on the client and server, both are performed, but user ID and password take precedence at the server. This

action is based on the CSIv2 specification requirements.

Authentication protocol support

Use this page to reference information regarding supported authentication protocols.

Authentication protocol support

Beginning with WebSphere Application Server Version 8.0, the WebSphere Application Server Version 8.0 servers only support the Common Secure Interoperability Version 2 (CSIv2) authentication protocol. Secure Authentication Service (SAS) is only supported between Version 6.0.*x* and previous version servers that have been federated in a Version 8.0 cell. The option to select between SAS, CSIv2, or both will only be made available in the administration console when a Version 6.0.x or previous release has been federated in a Version 8.0 cell.

In future releases, IBM will no longer ship or support the Secure Authentication Service (SAS) IIOP security protocol. It is recommended that you use the Common Secure Interoperability version 2 (CSIv2) protocol.

You can configure both protocols to work simultaneously between Version 6.0.*x* and previous version servers that have been federated in a Version 8.0 cell. If a server supports both protocols, it exports an interoperable object reference (IOR) that contains tagged components describing the configuration for SAS and CSIv2. If a client supports both protocols, it reads tagged components for both CSIv2 and SAS. If the client and server support both protocols, CSIv2 is used. However, if the server supports SAS (for example, the server is a previous WebSphere Application Server release) and the client supports both protocols, the client chooses SAS for this request.

Choose a protocol using the com.ibm.CSI.protocol property on the client side and configure this protocol through the administrative console on the server side.

You can configure both protocols to work simultaneously. If a server supports both protocols, it exports an interoperable object reference (IOR) that contains tagged components describing the configuration for SAS and CSIv2. If a client supports both protocols, it reads tagged components for both CSIv2 and SAS. If the client and the server support both protocols, CSIv2 is used. However, if the server supports SAS (for example, it is a previous WebSphere Application Server release) and the client supports both protocols, the client chooses SAS for this request.

Common Secure Interoperability Version 2 features

The following Common Secure Interoperability Version 2 (CSIv2) features are available in IBM WebSphere Application Server: message layer authentication, identity assertion, and security attribute propagation.

- Identity Assertion

  Supports a downstream server in accepting the client identity that is established on an upstream server, without having to authenticate again. The downstream server trusts the upstream server.
- Message Layer Authentication

  Authenticates credential information and sends that information across the network so that a receiving server can interpret it.
- Security attribute propagation

  Supports the use of the authorization token to propagate serialized Subject contents and PropagationToken contents with the request. You can propagate these objects using a pure client or a server login that adds custom objects to the Subject. Propagating security attributes prevents downstream logins from having to make user registry calls to look up these attributes.

  Propagating security attributes is also useful when the security attributes contain information that is only available at the time of authentication. This information cannot be located using the user registry on downstream servers.

Identity assertion to the downstream server

When a client authenticates to a server, the received credential is set. When the authorization engine checks the credential to determine whether access is permitted, it also sets the *invocation* credential . *Identity assertion* is the invocation credential that is asserted to the downstream server.

When a client authenticates to a server, the received credential is set. When the authorization engine checks the credential to determine whether access is permitted, it also sets the *invocation* credential so that if the Enterprise JavaBeans (EJB) method calls another EJB method that is located on other servers, the invocation credential can be the identity used to invoke the downstream method. Depending on the RunAs mode for the enterprise beans, the invocation credential is set as the originating client identity, the server identity, or a specified different identity. Regardless of the identity that is set, when identity assertion is enabled, it is the invocation credential that is asserted to the downstream server.

The invocation credential identity is sent to the downstream server in an identity token. In addition, the sending server identity, including the password or token, is sent in the client authentication token when basic authentication is enabled. The sending server identity is sent through a Secure Sockets Layer (SSL) client certification authentication when client certificate authentication is enabled. Basic authentication takes precedence over client certificate authentication.

Both identity tokens are needed by the receiving server to accept the asserted identity. The receiving server completes the following actions to accept the asserted identity:

- The server determines whether the sending server identity, sent with a basic authentication token or with an SSL client certificate, is on the trusted principal list of the receiving server. The server determines whether the sending server can send an identity token to the receiving server.
- After it is determined that the sending server is on the trusted list, the server authenticates the sending server to verify its identity.
- The server is authenticated by comparing the user ID and password from the sending server to the receiving server. If the credentials of the sending server are authenticated and on the trusted principal list, then the server proceeds to evaluate the identity token.
- The downstream server checks its defined user registry for the presence of the asserted user ID to gather additional credential information for authorization purposes (for example, group memberships). Thus, the downstream user registry must contain all of the asserted user IDs. Otherwise, identity assertion is not possible. In a stateful server, this action occurs once for the sending server and the receiving server pair where the identity tokens are the same. Subsequent requests are made through a session ID.

  Note: When the downstream server does not have a user registry with access to the asserted user IDs in its repository, do not use identity assertion because authorization checks will fail. By disabling identity assertion, the authorization checks on the downstream server are not needed.

Evaluation of the identity token consists of the following four identity formats that exist in an identity token:

- Principal name
- Distinguished name
- Certificate chain
- Anonymous identity

The product servers that receive authentication information typically support all four identity types. The sending server decides which one is chosen, based on how the original client authenticated. The existing type depends on how the client originally authenticates to the sending server. For example, if the client uses Secure Sockets Layer (SSL) client authentication to authenticate to the sending server, then the identity token sent to the downstream server contains the certificate chain. With this information, the receiving server can perform its own certificate chain mapping and interoperability is increased with other vendors and platforms.

After the identity format is understood and parsed, the identity maps to a credential. For an ITTPrincipal identity token, this identity maps one-to-one with the user ID fields.

For an ITTDistinguishedName identity token, the mapping depends on the user registry. For Lightweight Directory Access Protocol (LDAP), the configured search filter determines how the mapping occurs. For LocalOS, the first attribute of the distinguished name (DN), which is typically the same as the common name, maps to the user ID of the registry.

Identity assertion is only available using the Common Secure Interoperability Version 2 (CSIv2) protocol.

Note: There is a restriction for using identity assertion with KRB token to downstream. If you use identity assertion with Kerberos enabled, the identity assertion does not have the Kerberos authentication token (KRBAuthnToken) when going to downstream servers. It uses LTPA for authentication instead.

Identity assertions with trust validation

If you want an application or system provider to perform an identity assertion with trust validation, it can be accomplished by use of the Java Authentication and Authorization Service (JAAS) login framework, where trust validation is performed in one login module and credential creation in another. These two custom login modules are used to create a JAAS login configuration that performs a login to an identity assertion.

Two custom login module are required:

- A user-implemented trust association login module. This login module performs whatever trust verification the user requires. When trust is verified, the trust verification status and the login identity must be placed in a map in the share state of the login module to enable the credential creation login module to use that information. The map must be stored in the com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state property. State maps contain the following information:
  - com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trusted – set to `true`, if trusted, and `false`, if not trusted.
  - com.ibm.wsspi.security.common.auth.module.IdenityAssertionLoginModule.principal – contains the principal of the identity.
  - com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates – contains the certificate of the identity
- The com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule module performs the credential creation. It requires that the trust state information be in the login context's shared state. This login module is protected by the Java 2 security runtime permissions for the following:
  - com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.initialize
  - com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.login

IdentityAssertionLoginModule searches for the trust information in the shared state property, com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.state. This is a map that contains the trust status and the identity used to login. The map includes the following:

- com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.trusted – if set to `true` it is trusted, `false` if not trusted.
- com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.principal – if a principal is used, it contains the principal of the identity necessary to login.
- com.ibm.wsspi.security.common.auth.module.IdentityAssertionLoginModule.certificates – if a certificate is used, it contains an array of a certificate chain that includes the identity necessary to login.

A WSLoginFailedException is returned if the state, trust, or identity information is missing. The login module then performs a login of the identity. The subject now contains the new identity.

Message layer authentication

Defines the credential information and sends that information across the network so that a receiving server can interpret it.

When you send authentication information across the network using a token the transmission is considered message layer authentication because the data is sent with the message inside a service context.

A pure Java client uses Kerberos (KRB5) or basic authentication, or Generic Security Services Username Password (GSSUP), as the authentication mechanism to establish client identity.

However, a servlet can use either basic authentication (GSSUP) or the authentication mechanism of the server, Kerberos (KRB5) or Lightweight Third Party Authentication (LTPA), to send security information in the message layer. Use KRB5 or LTPA by authenticating or by mapping the basic authentication credentials to the security mechanism of the server.

The security token that is contained in a token-based credential is authentication mechanism-specific. The way that the token is interpreted is only known by the authentication mechanism. Therefore, each authentication mechanism has an object ID (OID) representing it. The OID and the client token are sent to the server, so that the server knows which mechanism to use when reading and validating the token. The following list contains the OIDs for each mechanism:

BasicAuth (GSSUP): oid:2.23.130.1.1.1
KRB5: OID: 1.2.840.113554.1.2.2
LTPA: oid:1.3.18.0.2.30.2
SWAM: No OID because it is not forwardable

Note: SWAM is deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release.

On the server, the authentication mechanisms can interpret the token and create a credential, or they can authenticate basic authentication data from the client, and create a credential. Either way, the created credential is the *received* credential that the authorization check uses to determine if the user has access to invoke the method. You can specify the authentication mechanism by using the following property on the client side:

- com.ibm.CORBA.authenticationTarget

Basic authentication (BasicAuth) and KRB5 are currently the only valid values. You can configure the server through the administrative console.

Note: When **perform basic authentication** is enabled, if the client is not similarly configured (and does not pass a credential such as a user ID and password).

Configuring authentication retries

Situations occur where you want a prompt to display again if you entered your user ID and password incorrectly or you want a method to retry when a particular error occurs back at the client. If you can correct the error by information at the client side, the system automatically performs a retry without the client seeing the failure, if the system is configured appropriately.

Some of these errors include:
- Entering a user ID and password that are not valid
- Having an expired credential on the server
- Failing to find the stateful session on the server

By default, authentication retries are enabled and perform three retries before returning the error to the client. Use the com.ibm.CORBA.authenticationRetryEnabled property (`True` or `False`) to enable or disable authentication retries. Use the com.ibm.CORBA.authenticationRetryCount property to specify the number of retry attempts.

Using Microsoft Active Directory for authentication

WebSphere Application Server supports the Microsoft Active Directory. Many installations use the Microsoft Active Directory as their primary component for managing user authentication and user data. Authenticating a user across multiple repositories or across a distributed Lightweight Directory Access Protocol (LDAP), such as a Microsoft Active Directory forest can be challenging. In any search of the whole registry, if there is more than one match at run time, authentication fails because ambiguous matches result.

About this task

User IDs are guaranteed to be unique within a single domain, but there is no automatic guarantee that a given user ID is unique across a tree or a forest. The following figure exemplifies the condition of a given user ID not being unique across a tree or forest.

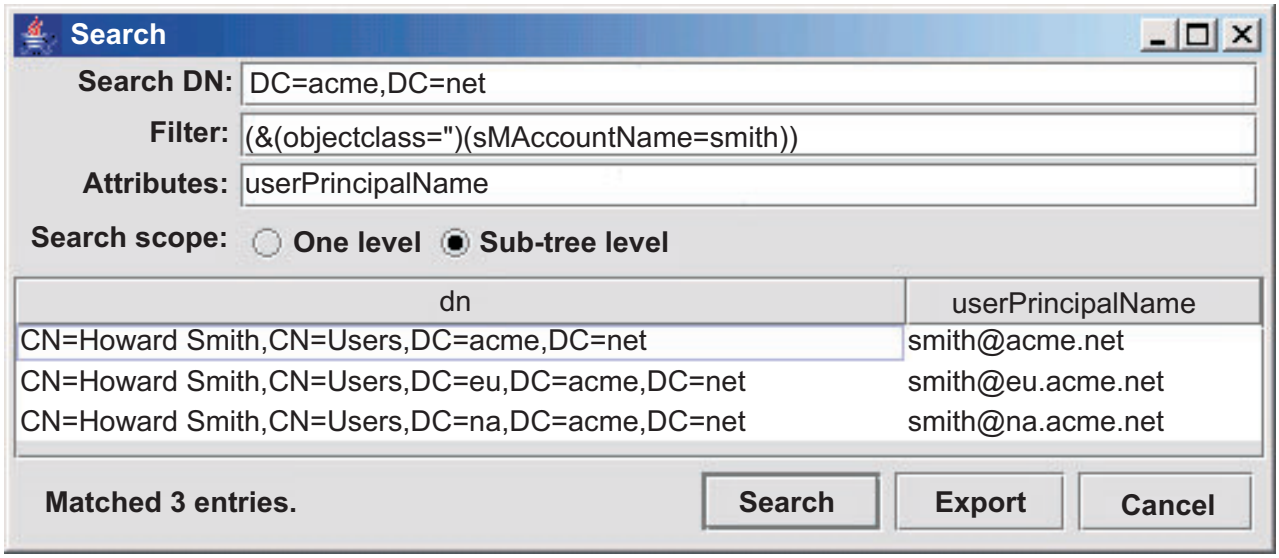


Figure 21. Forest search strategy.. Search illustration of a non-unique sAMAccountName across the entire forest.

Authenticating users across trees or forests can be a difficult task and the following steps should be performed.

Procedure

1. Analyze the Microsoft Active Directory construct that defines your installation. Your analysis can conclude with the following forms:
   - Single LDAP registry - Simple configuration.
   - Federated repository (a forest)- Typical configuration.
   - Merger of federated repositories (a merger of trees into a forest)- Less typical configuration
   - Combination of user and group forests - Rare configuration
2. Develop strategies for user look up that match your Microsoft Active Directory installation. Remember that user IDs are guaranteed to be unique within a single domain, but there is no automatic guarantee that a given user ID is unique across a tree or a forest.
3. Evaluate with testing to ensure that your authentication search strategies successfully authenticate users in your Microsoft Active Directory installation.

Results

You will be in the position to authenticate users with LDAP registries in a Microsoft Active Directory forest.

What to do next

gotcha: When you select any of these scenarios, consult appropriate Microsoft Active Directory information to completely understand any implications the scenarios might have on your configuation planning.

Authentication using Microsoft Active Directory

Many installations use the Microsoft Active Directory as their primary component for managing user authentication and user data. One portion of the Microsoft Active Directory provides a Lightweight Directory Access Protocol (LDAP) service. WebSphere Application Server supports LDAP and, therefore, WebSphere Application Server supports the Microsoft Active Directory.

While the Microsoft Active Directory is fully LDAP-compliant, it exposes LDAP information in ways that can make it difficult to obtain directory information for WebSphere Application Server.

WebSphere Application Server operates in a way that assumes that a single LDAP directory contains all the information necessary to operate. With complex Microsoft Active Directory configurations, this is not the case. WebSphere Application Server - Microsoft Active Directory installations must handle unique challenges because of the way data is spread throughout the domain controllers in a forest.

Microsoft Active Directory installations frequently incorporate the use of a forest. As such, security questions pertaining to user ID uniqueness, reliably obtaining user group information, and group membership spread across forests become important.

The following figure highlights a typical Microsoft Active Directory installation environment.

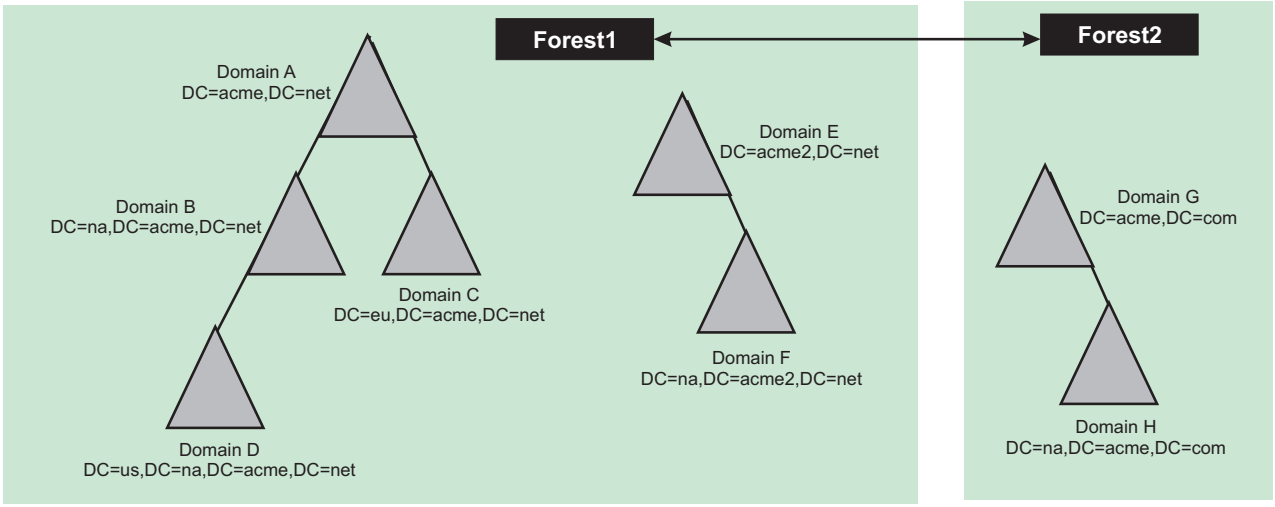


Figure 22. Microsoft Active Directory forests. An illustration of Microsoft Active Directory forests.

This figure illustrates two forests of one or more trees. A tree can contain one or more domains where the domain is the single atomic unit that forms the basis for the constructed environment. Each domain is made up of the primary domain components of the distinguished name (DN), for example, `dc=acme`, `dc=com`. A forest can extend trust to other forests (This trust is based on Kerberos.).

Microsoft Active Directory configurations with WebSphere Application Server

There can be a variety of Microsoft Active Directory configurations for WebSphere Application Server, which include:

- Simple configuration
- Typical configuration
- Less typical configurations
- Rare configurations

Simple configuration

The simplest configuration consists of a stand-alone LDAP registry representing a single domain. This configuration represents the closest fit between WebSphere Application Server and the Microsoft Active Directory. In this configuration, Microsoft Active Directory is supported through the WebSphere Application Server stand-alone LDAP user registry implementation. Alternatively, you can access this single Microsoft Active Directory domain through a federated repositories registry, which contains a single LDAP repository.

Typical Configuration

Beyond the simple single domain Microsoft Active Directory configuration, a typical Microsoft Active Directory configuration consists of a single tree in a forest where each branch of the tree is a domain. An example of this configuration, which consists of a single tree of four domains (A, B, C, D), is shown in the following example:

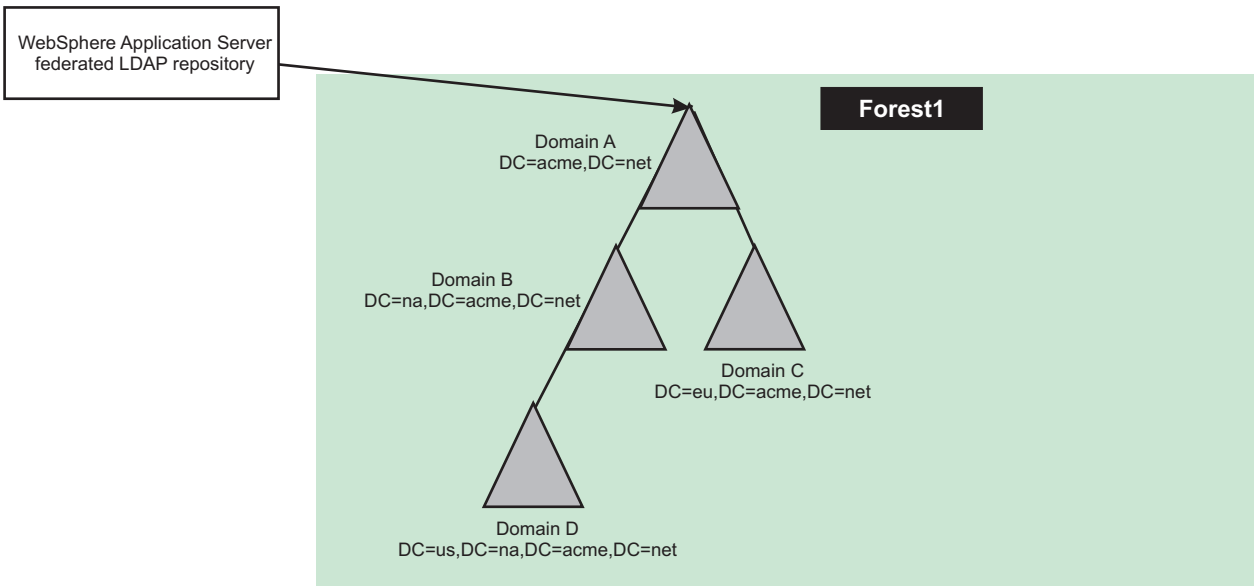


Figure 23. Typical forest configuration. An illustration of a typical forest configuration.

Configurations, such as this configuration, frequently have domains that are organized by geography or organizational unit. The WebSphere Application Server registry configuration that is necessary to use this "single tree" Microsoft Active Directory implementation needs to use the federated repositories. This configuration contains an LDAP registry to map entries from multiple individual user repositories into a single virtual repository. These configurations create a federated user repository with a single named realm and an LDAP subtree within the single repository. The root of the repository is mapped to a base entry

within the federated repository, which is the starting point within the hierarchical namespace of the virtual realm. LDAP searches in this configuration proceed with binding to the top domain object and following LDAP referrals.

gotcha: The stand-alone LDAP registry in WebSphere Application Server does not support LDAP referrals and cannot be used in a WebSphere Application Server - Microsoft Active Directory configuration.

Less typical configurations

Less typical WebSphere Application Server - Microsoft Active Directory configurations evolve from mergers of organizations units in a larger enterprise. Where a single forest of domains once served the enterprise, the merger of several new organizational units can add trees to the forest or even add more than a single forest to the environment. In this environment, the WebSphere Application Server LDAP configuration requires more careful design. You must use the federated repositories registry in such an environment with separate LDAP repositories mapped to the top of each tree in the forest. Again, if a Microsoft Active Directory tree exists under the top-level domain, LDAP referrals must be enabled for the LDAP registry. The forest resulting from a merger can look like the following figure:

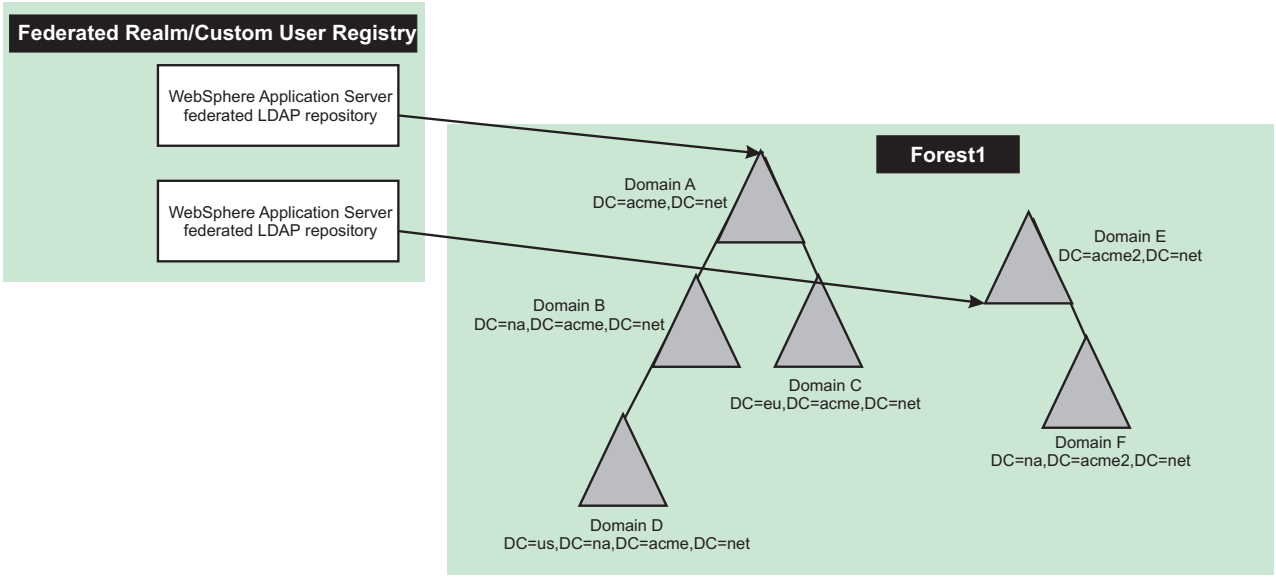


Figure 24. Less typical configurations. Less typical configurations that depict the merger of trees

Rare configurations

Rare configurations consist of Microsoft Active Directory domains that are configured where there is a combination of a user forest and a group forest. Users are imported as ForeignSecurityPrincipals objects in the group forest. The groups contain the distinguished names (DN) of the ForeignSecurityPrincipals objects as members.

In this form of configuration, direct group lookups do not occur. Lookups are relegated to a static group query across multiple registries. This configuration requires a custom user registry. However, WebSphere Application Server registries do not support this type of configuration. See the following figure.

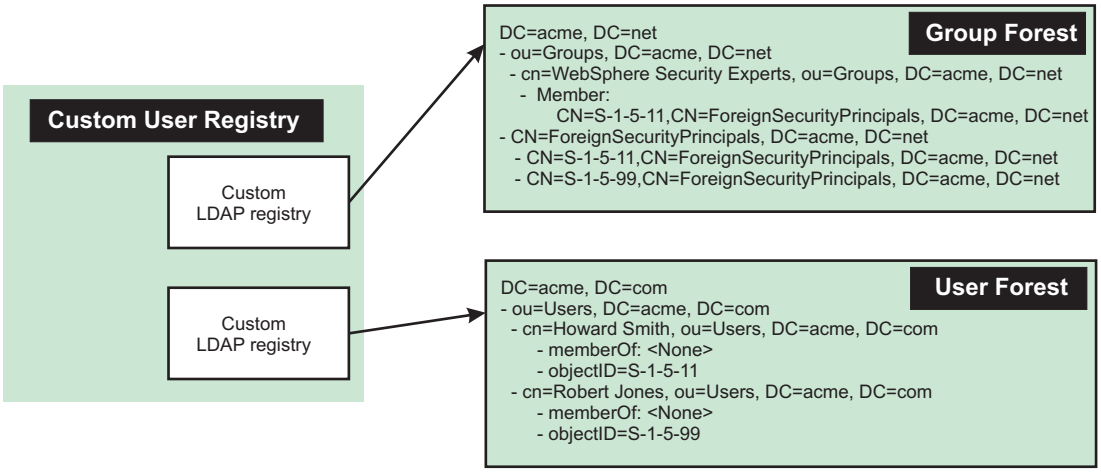


Figure 25. Resource model forest. An illustration of a resource model forest.

Using a Microsoft Active Directory forest as LDAP - user filter

Authenticating a user across multiple repositories, or across a distributed LDAP, such as a Microsoft Active Directory forest configuration can be challenging. In any search of the whole registry, authentication fails if there is more than one match at run time because ambiguous matches result. In multiple Microsoft Active Directory domain environment, the WebSphere Application Server administrator must consider that the default unique ID in the Microsoft Active Directory is the sAMAccountName attribute of a user. User IDs are guaranteed to be unique within a single domain, but it is not possible to guarantee that a given user ID is unique across a tree or a forest. See the topic, "Authenticating users with LDAP registries in a Microsoft Active Directory forest" to understand how to search for user IDs within a Microsoft Active Directory forest using the sAMAccountName attribute of a user.

gotcha: Before selecting any of these scenarios, consult appropriate Microsoft Active Directory information to completely understand any implications the scenarios might have on your configuation planning.

Groups spanning domains with Microsoft Active Directory

The domains and forests functional levels of the Microsoft Active Directory control which configurations are available for use. How you configure Microsoft Active Directory affects how group membership is determined within WebSphere Application Server. Using groups to configure your Microsoft Active Directory installation with the product allows flexible management.

A breakdown follows of applicable functional levels that apply to a Microsoft Active Directory installation with the product.

- Domain Functional Levels
  - Native
    - Supported by Windows Server 2008 and Windows Server 2008 R2
    - Default in Windows 2008

  You must use native domain functional levels to support group nesting, and universal groups. Forest functional levels do not directly affect group membership. The Windows 2008 operating system is the exception.
- Forest Functional Levels
  - Windows Server 2008 or Windows Server 2008 R2
    - All domains operate at the Windows Server 2008 **domain functional level**.

If the forest functional level is set to Windows Server 2008, then that also makes the domain functional level for all domains to be Windows Server 2008 Native level, which adds to the group nesting and Universal groups features to Microsoft Active Directory.

Microsoft Active Directory groups

In a domain, Microsoft Active Directory provides support for different types of groups and group scopes. Groups in Microsoft Active Directory are containers with other objects within them as members. Those objects can be user objects, other group objects, which is group nesting, and other objects types, such as computers. The group type determines the type of task that you manage with the group. The group scope determines whether the group can have members from multiple domains or a single domain. In summary:

- Groups are typically a collection of user accounts.
- Members receive permission given to groups.
- Users can be members of multiple groups.
- Groups can be members of other groups, which are nested groups.

gotcha: In WebSphere Application Server, security roles of the individual, which map to application permissions or authorizations, must be bound to either users or groups at application deployment time. From an administrative point of view, it is preferable to assign permissions once for a group instead of assigning permissions repeatedly for each user account. Then the ability to act in a given role is under the control of the directory administrator, instead of the WebSphere administrator. Because the job of the directory administrator is to create and delete users, change group memberships for users, and other tasks, this approach is generally the correct division of responsibilities.

Group types determine how the group is used. The Microsoft Active Directory group types are:

- **Security groups:** Microsoft Active Directory uses security groups for granting permissions to gain access to resources.
- **Distribution groups:** Distribution groups are used by Windows-based applications as lists for nonsecurity-related functions. Distribution groups are used for sending email messages to groups of users. You cannot grant Windows permissions to security groups.

Although WebSphere Application Server can use either type of group, security groups are typically bound to WebSphere Application Server security roles.

Group scopes describe which type of objects can be arranged together within a group. Group nesting describes when one group is a member of other groups. The Microsoft Active Directory group scopes are:

- **Domain local group:**
  - **Windows usage:** Members of this group can come from any domain, but can access Windows resources only in the local domain. Use this scope to grant permissions to domain resources that are located in the same domain in which you created the domain local group. Domain local groups can exist in all mixed, native, and interim functional level of domains and forests.
  - **Restriction:** You cannot define group nesting in a domain local group. A domain local group cannot be a member of another domain local group or any other group in the same domain.
  - **WebSphere usage:** Users are not typically placed in domain - local groups due to these restrictions. WebSphere Application Server security roles are not typically bound to domain local groups.
- **Global Group:**
  - **Windows usage:** Members of this group originate from a local domain, but can access Windows resources in any domain. The global group is used to organize users who share similar Windows network access requirements. You can add members only from the domain in which the global group is created. You can use this group to assign permissions to gain access to Windows resources that are located in any domain in the domain, tree, or forest.

You can group users with similar function under global scope and give permission to access a Windows resource, such as a printer or shared folder and files, that is available in local or another domain in the same forest. You can use global groups to grant permission to gain access to Windows resources that are located in any domain in a single forest as their memberships are limited. You can add user accounts and global groups only from the domain in which global group is created.

Nesting is possible for global groups within other groups as you can add a global group into another global group from any domain. Members of a global group can be members of a domain - local group. Global groups exist in all mixed, native, and interim functional levels of domains and forests.

WebSphere Application Server usage: Global groups are visible on every domain controller, but memberships are only visible for local users. That is, you can see your group memberships only if you query your home domain controller. A global group should contain groups of users. Global groups are intended to be included in universal groups.

- **Universal Group:**
  - **Windows usage:** Members in this group can come from any domain and access Windows resources in multiple domains. Universal group memberships are not limited like global groups. All domain user accounts and groups can be members of a universal group.
  - **Restrictions:**
    - Universal groups are available when the domain is at a Windows mixed functional level.
    - It can be expensive to replicate this data across the forest. Group definitions and deletions are relatively rare compared to the equivalent user actions, and nested group membership changes are typically rare compared to memberships of users within groups,

      gotcha: Consult appropriate Microsoft Active Directory information concerning any implications of replicating data across forests.
  - **WebSphere usage:**
    - Universal Groups and their memberships are visible on every domain controller in the forest.
    - Universal groups are also visible when using the Global Catalog. To be useful, all user objects must be directly in the universal group,

  Universal group guidelines

  1. Assign permissions to universal groups for Windows resources in any domain in the network.
  2. Use universal groups only when their membership is static. Changes in membership can cause excessive network traffic between domain controllers. Membership of universal groups can be replicated to many domain controllers.
  3. Add global groups from several domains to a universal group.
  4. Assign permissions for access to a Windows resource to the universal group and for use by WebSphere Application Server group membership resolution across multiple domains.
  5. Use a universal group in the same way as a domain local group to assign resource permissions.

gotcha: When you select any of these scenarios, consult the appropriate Microsoft Active Directory information to completely understand any implications the scenarios might have on your configuation planning.

Microsoft Active Directory Global Catalog

A *Global Catalog* is a Global Catalog Server. A Global Catalog holds a full set of attributes for the domain in which it resides and a subset of attributes for all objects in the Microsoft Active Directory Forest. The primary two functions of a Global Catalog within the Microsoft Active Directory are logon capability and Microsoft Active Directory queries.

A Global Catalog in a Microsoft Active Directory installation with the product is a single Lightweight Directory Access Protocol (LDAP) repository that contains a subset of user information from all the domains in the forest. This information includes user IDs, authentication information, and groups, but not all the group information.

You can use the Global Catalog on any domain controller in the forest, even in subdomains. The Global Catalog is a solution to the WebSphere Application Server limitation of a "single registry". There are limitations to the Global Catalog. Users from the local domain controller contain group "memberOf" information. Users from a foreign domain controller contain limited "memberOf" information because the global group information is not replicated to every domain controller.

Nested global groups in universal groups

This is a typical structure of group membership and consists of the following characteristics:

- Users are distributed across domain controllers in a forest containing multiple domain controllers.
- Users are defined in global groups within their own local domain controller.
- A universal group contains the global groups, which reflects a Java Platform Enterprise Edition (Java EE) role that maps to a set of users spread across *multiple* domain controllers.

The following figure illustrates nested global groups in universal groups.

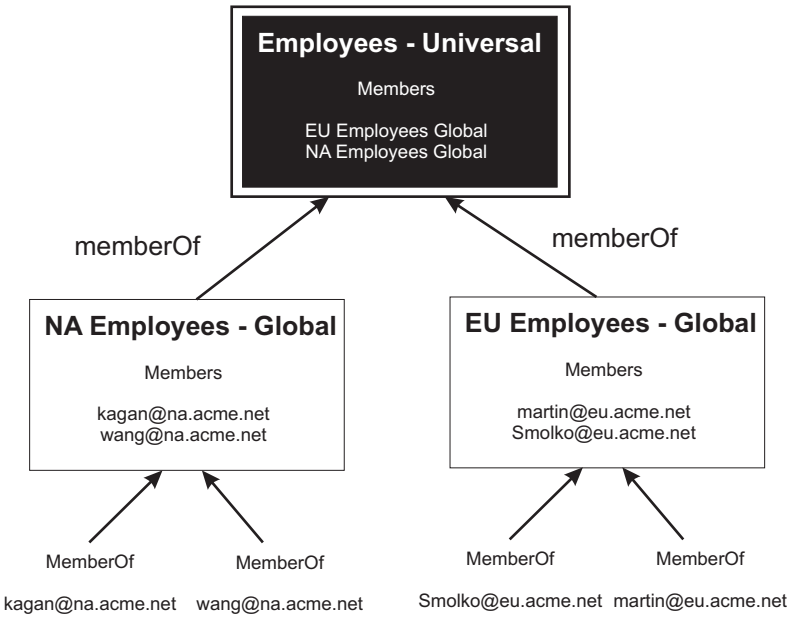


Figure 26. Nested global groups in universal groups. This figure illustrates nested global groups in universal groups.

It is a challenge to develop methods of configuring WebSphere Application Server to be able to find users and their group memberships when the information is spread across multiple domain controllers. One method requires that WebSphere Application Server follow LDAP referrals to find the home domain controller for each user and that WebSphere Application Server perform nested group queries.

gotcha: This approach does not use the Global Catalog.

Another method and the simplest approach has universal groups that contain users and uses a Global Catalog, which requires using referrals. The figure that follows illustrates this method.

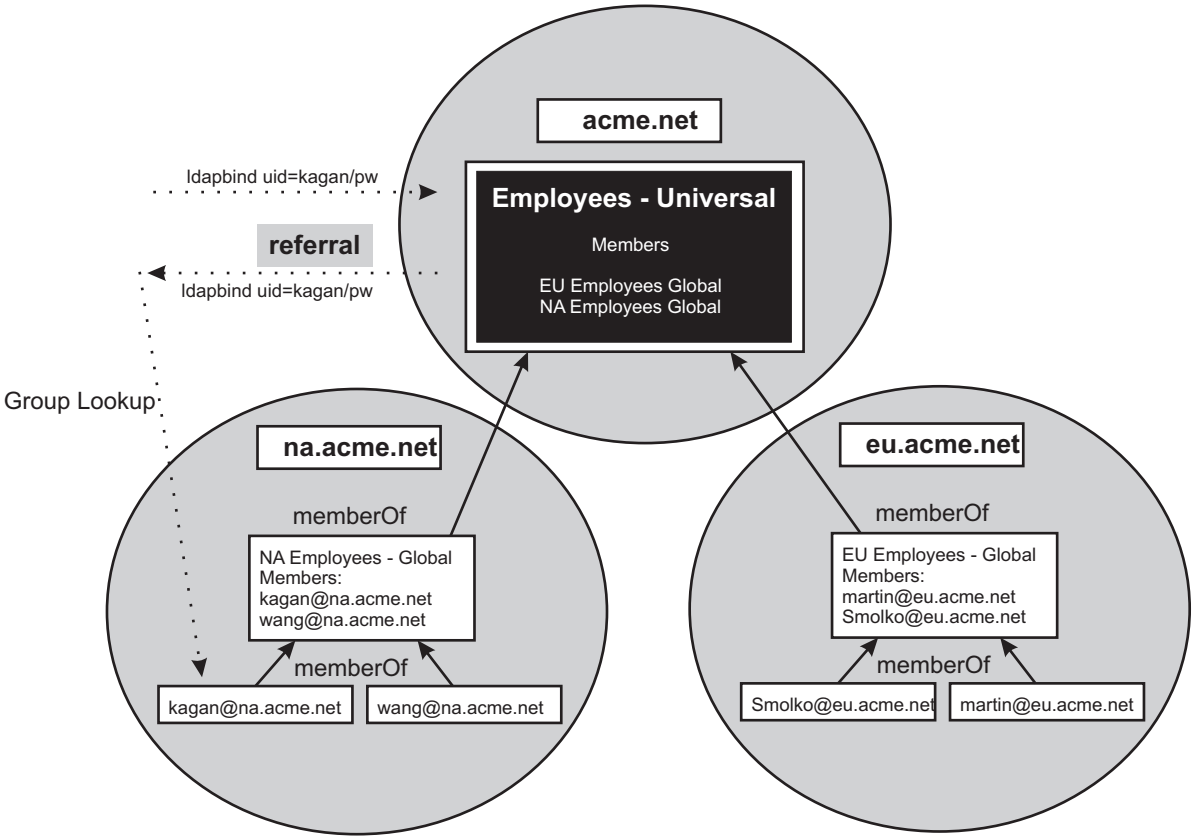


Figure 27. Locating group memberships. This figure illustrates the process of locating group memberships.

A variation on this method is to not use universal groups. You can use this approach when universal groups are not available.

gotcha: This approach does not use the Global Catalog.

You might consider using the Microsoft Active Directory Global Catalog as the WebSphere Application Server registry. There are three scenarios; however, the first two scenarios demonstrate how failures occur.

1. If you configure WebSphere Application Server to use Global Catalog as its LDAP registry and follow referrals, then individual users are visible in each domain controller. Because a user must exist only once in the registry, all logins fail.
2. If you configure WebSphere Application Server to use Global Catalog as its LDAP registry and do not follow referrals and the individual users are within global groups, then group membership is incomplete. See the following figure, which illustrates this limitation.

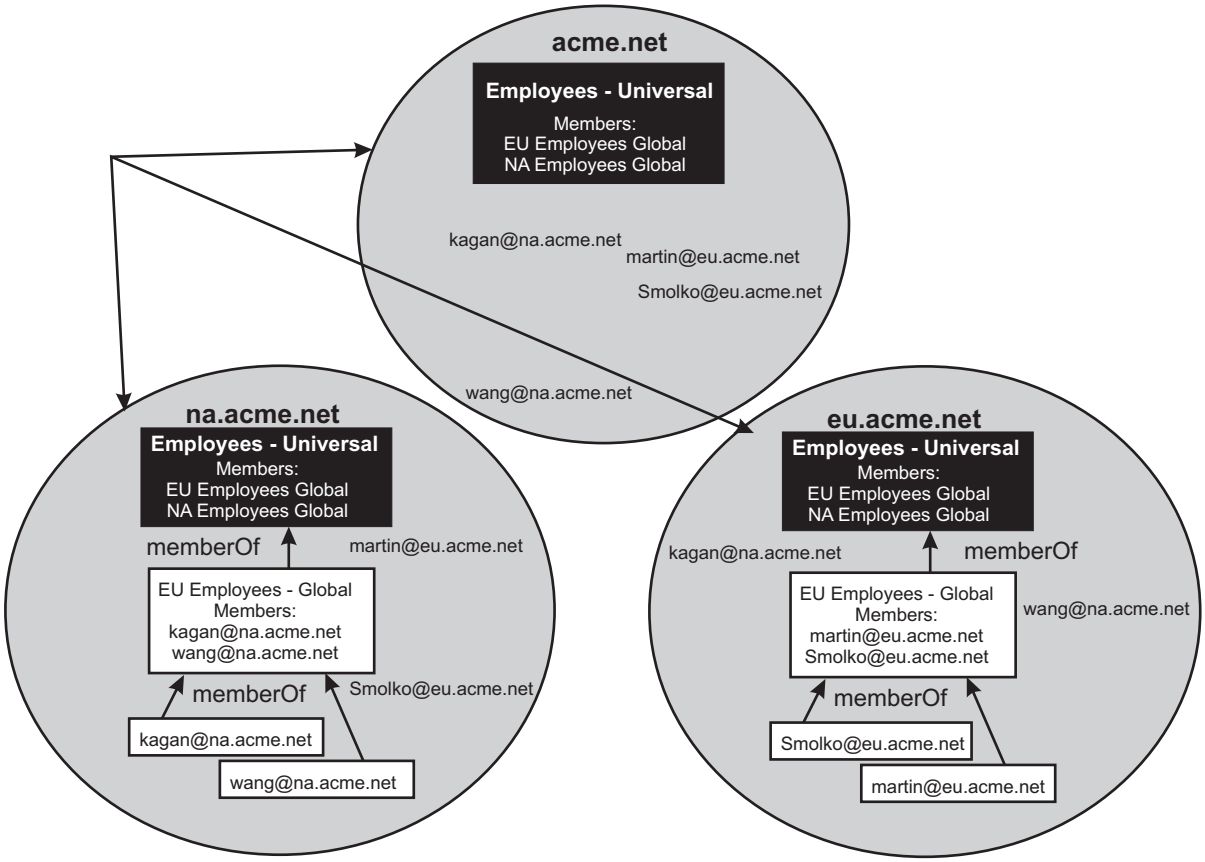


Figure 28. Global catalog (without using referrals). An illustration of a Global Catalog without using referrals

3. When you configure WebSphere Application Server to use Global Catalog as its LDAP registry, do not follow referrals, and users are directly contained within universal global groups, then group membership is complete.

gotcha: When you select any of these scenarios, consult appropriate Microsoft Active Directory information to completely understand any implications the scenarios might have on your configuation planning.

Options for finding group membership within a Microsoft Active Directory forest

Locating and finding group membership with the Microsoft Active Directory forest is necessary for authenticating users. There are several ways to approach finding group membership within the Microsoft Active Directory forest.

The following figure depicts an example of group membership with the Microsoft Active Directory forest. This figure is used to explain ways to find group membership.

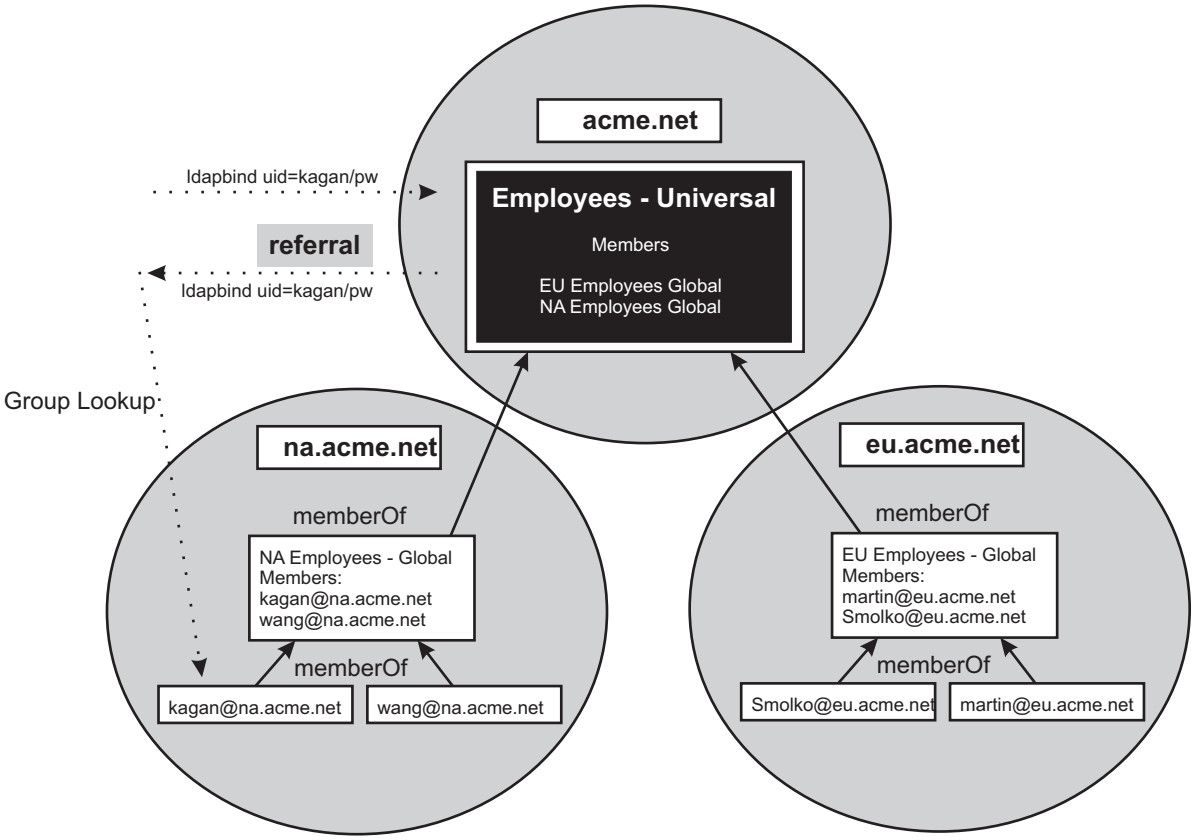


Figure 29. Finding group membership.. An illustration of ways to find group membership.

- **Option 1** does not use nested groups, and the following steps describe the process of locating group membership using a hypothetical organizational structure.
  - Create a global group of **NA employees**.
  - Create a global group of **EU employees**.
  - Map the Java Platform Enterprise Edition (Java EE) role to **NA employees** + **EU employees**. This mapping can become unmanageable if there are too many sub domains
  - Enable referrals.

    In WebSphere Application Server Version 6.1, use federated repositories, specifically:
    - Use a federated realm.
    - Add the Microsoft Active Directory top-level domain controller to the repository. Do not add sub-domain controllers. Doing this results in multiple matches when searches for user IDs occur. The multiple matches cause user logins to fail.
    - Select "**Support referrals to other LDAP servers**" = "follow".
- **Option 2** uses universal groups.
  - Put individual users into the universal group, **Employees**.

    Requirements:
    - The Windows 2003 Native domain functional levels is required.
    - Userids must be directly contained within universal groups.
  - Map Java EE role to **Employees**.
  - Connect to any global catalog in the forest.

Tip: This option reduces the amount of directory lookup traffic. WebSphere Application Server does not have to follow all the referrals across the directory tree. That is, each domain controller can fully resolve the group information locally.

- **Option 3** uses nested groups.
  - Create the universal group, **Employees**.
  - Create **NA Employees** and **EU Employees** as global groups and make them members in the **Employees** universal group.

    Requirements: Windows Native Domain functional levels.
  - Map Java JEE role to "Employees".
  - Enable referrals.

    For WebSphere Application Server Version 6.1, use federated repositories, specifically:
    - Use a federated realm.
    - Add the Active Directory top-level domain controller to the repository. Do not add sub-domain controllers, as this will result in multiple matches when searches for userids occur, and logins will fail.
    - Select "Support referrals to other LDAP servers" = "follow".
  - Enable nested groups.

Tip: This option offers the optimal approach when using WebSphere Application Server Versions 6.1 or later. Before WebSphere Application Server version 6.1, referrals were not officially supported.

Summary

The following table summarizes how to find group membership within a Microsoft Active Directory forest.

Table 122. Finding group membership.. The following table identifies group membership levels supported in a Microsoft Active Directory forest.

Group Membership	Map Java EE Roles To	Bind to Which LDAP	Enable	Supported in WebSphere Application Server Version	Comments
Global Groups	Collection of global groups	Top domain controller using port 389/636	Referrals	• Federated repositories in WebSphere Application Server	
Universal groups	Universal groups	Any Global catalog, using port 3268		All	
Global groups in universal groups	Universal groups	Top domain controller using port 389/636	referrals, nesting	• Federated repositories in WebSphere Application Server	Cannot use Windows mixed domain functional level

Configuring to use objectCategory attribute

A federated repository uses the `objectCategory` attribute by default for Active Directory user search filters. You can ensure that the federated repository is configured to use the objectCategory attribute. For example, the federated repositories configuration file, `wimconfig.xml`, should be as shown in the following example:

```
<supportedLDAPEntryType name="user" searchFilter="(objectCategory=user)"...>
<supportedLDAPEntryType name="Group" searchFilter="(objectCategory=Group)"...>
```

Configure the user filter and group filter (advanced properties) like the following example:

```
User Filter: (&(sAMAccountName=%v)(objectCategory=user))
Group Filter: (&cn=%v)(objectCategory=group)
```

Follow the following instructions from the administrative console to complete the search filter with the objectCategory attribute.

1. Click **Security**.
2. Select **Global security**.
3. Under Available realm definitions, use the drop-down list to select **Federated repositories**.
4. Click **Configure**.
5. Under Related items, click **Manage repositories**.
6. Select **Forest > LDAP entity types > PersonAccount**. Under General Properties, find the **Search filter** box.
7. Fill in the search filter.

   ```
   (objectCategory=user)
   ```

gotcha: When you select any of these scenarios to use, consult the appropriate Microsoft Active Directory information to completely understand any implications the scenarios might have on your configuation planning.

Authenticating users with LDAP registries in a Microsoft Active Directory forest

Authenticating a user across multiple repositories, or across a distributed Lightweight Directory Access Protocol (LDAP) repository, such as a Microsoft Active Directory forest can be challenging. In any search of the whole user registry, if there is more than one match at run time, authentication fails because of ambiguous match results.

Before you begin

In any multiple Microsoft Active Directory domain environment, the WebSphere Application Server administrator must consider that the default unique ID in the Microsoft Active Directory is the sAMAccountName attribute of a user.

About this task

User IDs are guaranteed to be unique within a single domain. However they are not guaranteed across a tree or a forest. For example, suppose the user ID, *smith*, is added in the forest and in each subdomain. The search for `sAMAccountName=smith` returns three matches. WebSphere Application Server does not authenticate this user when there is more than one possible match in the registry.

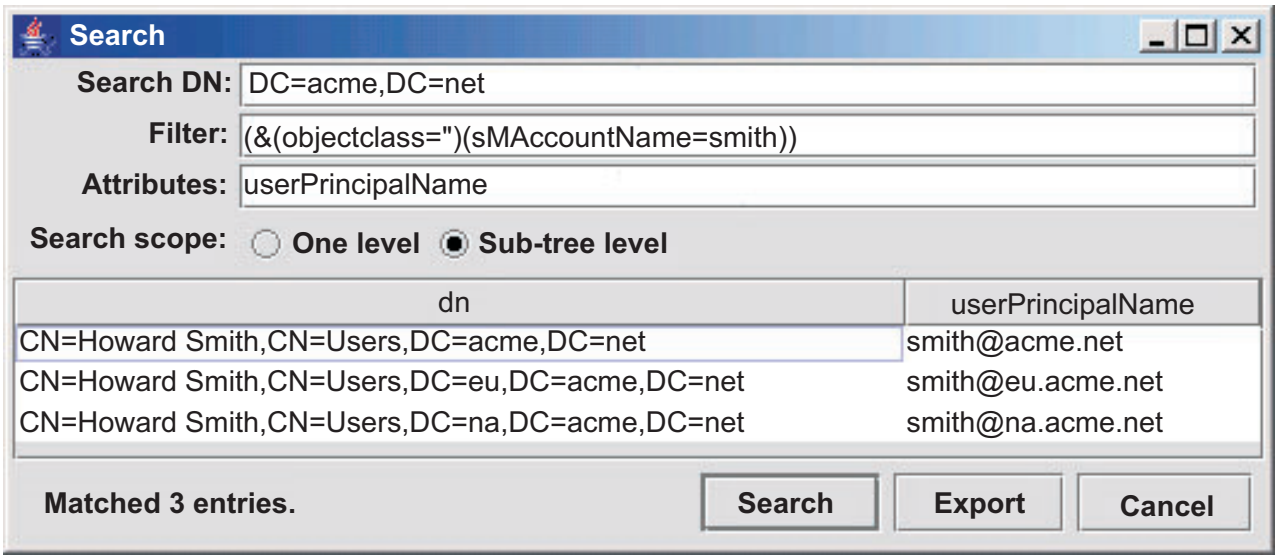


Figure 30. Forest search strategy.. Search illustration of a non-unique sAMAccountName across the entire forest.

You can mitigate this condition by changing the user filter to be based on the userPrincipalName attribute of the user, which is unique across the forest, instead of being based on their sAMAccountName attribute. However, users must then know to log in using their userPrincipalName, which they might not know.

The specific procedure to establish a user filter on a LDAP user registry depends on the type of LDAP registry. The following examples illustrate a procedure for a stand-alone LDAP registry and a procedure for a federated repository registry.

Procedure

1. **Establish a user filter on a stand-alone LDAP registry:** You can set the user filter on the Advance Lightweight Access Protocol (LDAP) user registry settings page to search for userPrincipalName instead of sAMAccountName value.

   For example:

   ```
   (&(objectClass=user)(userPrincipalName=%w))
   ```

2. **Establish a user filter on a federated repositories registry:** You can change the log-in property in the LDAP repository to `uid;cn`, for example, by using the administrative console.
   a. Click **Security**.
   b. Select **Global security**.
   c. Under Available realm definitions, use the drop-down list to select **Federated repositories**.
   d. Click **Configure**.
   e. Under Related items, click **Manage repositories**.
   f. Under General Properties, add the following information:

   Repository identifier
   forest

   Directory type
   Microsoft WIndows Server 2003 Active Directory

   Primary host name
   forest.acme.net

   Port 389

   Failover server used when primary is not available
   Delete

Bind distinguished name
cn=wasbind, CN=Users, DC=ib

Bind password
****

Login properties
uid;cn

3. Click **OK** and **Save** to save the changes to the master configuration.
4. Locate the `{WAS_HOME}\profiles\{profileName}\config\cells\{cellName}\wim\config\wimconfig.xml` or `profile_root/conf/cells/<cell>/wim/config/wimconfig.xml` file in the deployment manager configuration.
5. Edit the `wimconfig.xml` file.
   a. Find the `<config:attributeConfiguration>` attribute in the file.
   b. Add the following lines:

```
<config:attributes name="userPrincipalName" propertyName="cn">
<config:entityTypes>PersonAccount</config:entityTypes>
</config:attributes>
```

6. Save the `wimconfig.xml` file.
7. Run the `profile_root/bin/syncNode.bat` or `profile_root/syncNode.bar/sh` script on all of the nodes in the configuration.

Results

gotcha: When you select any of these scenarios, consult appropriate Microsoft Active Directory information to completely understand any implications the scenarios might have on your configuation planning.

Authorizing access to resources

WebSphere Application Server provides many different methods for authorizing accessing resources. For example, you can assign roles to users and configure a built-in or external authorization provider.

About this task

You can create an application, an Enterprise JavaBeans (EJB) module, or a web module and secure them using assembly tools.

To authorize user or group access to resources, read the following articles:

Procedure

1. Secure you application during assembly and deployment. For more information on how to create a secure application using an assembly tool, such as the IBM Rational Application Developer, see the information about securing applications during assembly and deployment.
2. Authorize access to Java Platform, Enterprise Edition (Java EE) resources. WebSphere Application Server supports authorization that is based on the Java Authorization Contract for Containers (JACC) specification in addition to the default authorization. When security is enabled in WebSphere Application Server, the default authorization is used unless a JACC provider is specified. For more information, see “Authorization providers” on page 1627.
3. Authorize access to administrative resources. You can assign users and groups to predefined administrative roles such as the monitor, configurator, operator, administrator, auditor and iscadmins roles. These roles determine which tasks a user can perform in the administrative console. For more information, see “Authorizing access to administrative roles” on page 1679.

What to do next

After authorizing access to resources, configure the Application Server for secure communication. For more information, see “Securing communications” on page 1700.

Authorization technology

Authorization information determines whether a user or group has the necessary privileges to access resources.

WebSphere Application Server supports many authorization technologies including the following:

- Authorization involving the web container and Java Platform, Enterprise Edition (Java EE) technology
- Authorization involving an enterprise bean application and Java EE technology
- Authorization involving web services and Java EE technology
- Java Message Service (JMS)
- Java Authorization Contract for Containers (JACC)

  WebSphere Application Server supports both a default authorization provider and an authorization provider that is based on the Java Authorization Contract for Containers (JACC) specification. The JACC-based authorization provider enables third-party security providers to handle the Java EE authorization. For more information, see “JACC support in WebSphere Application Server” on page 1627.
- Java Authentication and Authorization Service (JAAS)

  For more information, see “Java Authentication and Authorization Service” on page 1516.
- Java 2 security

  For more information, see “Java 2 security” on page 1176.
- Naming and administrative authorization
- Pluggable authorization

  WebSphere Application Server supports an authorization infrastructure that enables you to plug in an external authorization provider. For more information, see “Enabling an external JACC provider” on page 1649.

Administrative roles and naming service authorization

WebSphere Application Server extends the Java Platform, Enterprise Edition (Java EE) security role-based access control to protect the product administrative and naming subsystems.

Administrative roles

A number of administrative roles are defined to provide the degrees of authority that are needed to perform certain WebSphere Application Server administrative functions from either the administrative console or the system management scripting interface called wsadmin. The authorization policy is only enforced when administrative security is enabled. The following table describes the administrative roles:

Table 123. Administrative roles that are available through the administrative console and wsadmin.

This table lists administrative roles that are available through the administrative console and wsadmin.

Role	Description
Monitor	An individual or group that uses the monitor role has the least amount of privileges. A monitor can complete the following tasks: • View the WebSphere Application Server configuration. • View the current state of the Application Server.

Table 123. Administrative roles that are available through the administrative console and wsadmin (continued).

This table lists administrative roles that are available through the administrative console and wsadmin.

Role	Description
Configurator	An individual or group that uses the configurator role has the monitor privilege plus the ability to change the WebSphere Application Server configuration. The configurator can perform all the day-to-day configuration tasks. For example, a configurator can complete the following tasks: • Create a resource. • Map an application server. • Install and uninstall an application. • Deploy an application. • Assign users and groups-to-role mapping for applications. • Set up Java 2 security permissions for applications. • Customize the Common Secure Interoperability Version 2 (CSIv2), Secure Authentication Service (SAS), and Secure Sockets Layer (SSL) configurations. **Important:** SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.
Operator	An individual or group that uses the operator role has monitor privileges plus ability to change the runtime state. For example, an operator can complete the following tasks: • Stop and start the server. • Monitor the server status in the administrative console.
Administrator	An individual or group that uses the administrator role has the operator and configurator privileges plus additional privileges that are granted solely to the administrator role. For example, an administrator can complete the following tasks: • Modify the server user ID and password. • Configure authentication and authorization mechanisms. • Enable or disable administrative security. **Note:** In previous releases of WebSphere Application Server, the **Enable administrative security** option is known as the **Enable global security** option. • Enforce Java 2 security using the **Use Java 2 security to restrict application access to local resources** option. • Change the Lightweight Third Party Authentication (LTPA) password and generate keys. • Create, update, or delete users in the federated repositories configuration. • Create, update, or delete groups in the federated repositories configuration. **Note:** An administrator cannot map users and groups to the administrator roles.
Adminsecuritymanager	Only users who are granted this role can map users to administrative roles. Also, when fine-grained administrative security is used, only users who are granted this role can manage authorization groups. See “Administrative roles” on page 1623 for more information.
Deployer	Users who are granted this role can perform both configuration actions and runtime operations on applications.

Table 123. Administrative roles that are available through the administrative console and wsadmin (continued).

This table lists administrative roles that are available through the administrative console and wsadmin.

Role	Description
Auditor	Users granted this role can view and modify the configuration settings for the security auditing subsystem. For example, a user with the auditor role can complete the following tasks: • Enable and disable the security auditing subsystem. • Select the event factory implementation to be used with the event factory plug-in point. • Select and configure the service provide, or emitter. or both to be used with the service provider plug-in point. • Set the audit policy that describes the behavior of the application server in the event of an error with the security auditing subsystem. • Define which security events are to be audited. The auditor role includes the monitor role. This allows the auditor to view but not change the rest of the security configuration.

Table 124. Additional administrative role that is available through the administrative console.

This table lists an additional administrative role that is available through the administrative console.

Role	Description
iscadmins	This role is only available for administrative console users and not for wsadmin users. Users who are granted this role have administrator privileges for managing users and groups in the federated respositories. For example, a user of the iscadmins role can complete the following tasks: • Create, update, or delete users in the federated repositories configuration. • Create, update, or delete groups in the federated repositories configuration.

Table 125. Additional administrative role that is available through wsadmin.

This table lists an additional administrative role that is available through the administrative console.

Role	Description
Deployer	This role is only available for wsadmin users and not for administrative console users. Users who are granted this role can perform both configuration actions and run-time operations on applications.

When administrative security is enabled, the administrative subsystem role-based access control is enforced. The administrative subsystem includes the security server, the administrative console, the wsadmin scripting tool, and all the Java Management Extensions (JMX) MBeans. When administrative security is enabled, both the administrative console and the administrative scripting tool require users to provide the required authentication data. Moreover, the administrative console is designed so the control functions that display on the pages are adjusted, according to the security roles that a user has. For example, a user who has only the monitor role can see only the non-sensitive configuration data. A user with the operator role can change the system state.

When you are changing registries (for example, from a federated repository to LDAP), make sure you remove the information that pertains to the previously configured registry for console users and console groups.

When administrative security is enabled, WebSphere Application Servers run under the server identity that is defined under the active user registry configuration. Although it is not shown on the administrative console and in other tools, a special Server subject is mapped to the administrator role. The WebSphere Application Server runtime code, which runs under the server identity, requires authorization to runtime operations. If no other user is assigned administrative roles, you can log into the administrative console or to the wsadmin scripting tool using the server identity to perform administrative operations and to assign

other users or groups to administrative roles. Because the server identity is assigned to the administrative role by default, the administrative security policy requires the administrative role to perform the following operations:

- Change server ID and server password
- Enable or disable WebSphere Application Serveradministrative security
- Enforce Java 2 security using the **Use Java 2 security to restrict application access to local resources** option.
- Change the LTPA password or generate keys
- Assign users and groups to administrative roles

Primary administrative user name

The Version 6.1 release of WebSphere Application Server and subsequent releases require an administrative user, distinguished from the server user identity, to improve auditability of administrative actions. The user name specifies a user with administrative privileges that is defined in the local operating system.

Server user identity

The Version 6.1 release of WebSphere Application Server and subsequent releases distinguish the server identity from the administrative user identity to improve auditability. The server user identity is used for authenticating server-to-server communications.

Internal server ID

The internal server ID enables the automatic generation of the user identity for server-to-server authentication. Automatic generation of the server identity supports improved auditability for cells only for Version 6.1 or later nodes. In the Version 6.1 release of WebSphere Application Server, you can save the internally-generated server ID because the Security WebSphere Common Configuration Model (WCCM) model contains a new tag, internalServerId. You do not need to specify a server user ID and a password during security configuration except in a mixed-cell environment. An internally-generated server ID adds a further level of protection to the server environment because the server password is not exposed as it is in releases prior to Version 6.1. However, to maintain backwards compatibility, you must specify the server user ID if you use earlier versions of WebSphere Application Server.

When enabling security, you can assign one or more users and groups to naming roles. For more information, see Assigning users to naming roles. However, before assigning users to naming roles, configure the active user registry. User and group validation depends on the active user registry. For more information, see Configuring user registries.

Special subject

In addition to mapping users or groups, you can map a *special-subject* to the administrative roles. A special-subject is a generalization of a particular class of users. The AllAuthenticated or the AllAuhenticatedInTrustedRealms (when cross realm is involved) special subjects mean that the access check of the administrative role ensures that the user making the request is at least authenticated. The Everyone special subject means that anyone, authenticated or not, can perform the action as if security is not enabled.

Naming service authorization

CosNaming security offers increased granularity of security control over CosNaming functions. CosNaming functions are available on CosNaming servers such as the WebSphere Application Server. These functions affect the content of the WebSphere Application Server name space. Generally, you have two ways in

which client programs result in CosNaming calls. The first is through the Java Naming and Directory Interface (JNDI) call. The second is with common object request broker architecture (CORBA) clients invoking CosNaming methods directly.

Four security roles are introduced :

- CosNamingRead
- CosNamingWrite
- CosNamingCreate
- CosNamingDelete

The roles have authority levels from low to high:

CosNamingRead
: You can query the WebSphere Application Server name space, using, for example, the JNDI lookup method. The special-subject, Everyone, is the default policy for this role.

CosNamingWrite
: You can perform write operations such as JNDI bind, rebind, or unbind, and CosNamingRead operations. As a default policy, Subjects are not assigned this role.

CosNamingCreate
: You can create new objects in the name space through such operations as JNDI createSubcontext and CosNamingWrite operations. As a default policy, Subjects are not assigned this role.

CosNamingDelete
: You can destroy objects in the name space, for example using the JNDI destroySubcontext method and CosNamingCreate operations. As a default policy, Subjects are not assigned this role.

A Server special-subject is assigned to all of the four CosNaming roles by default. The Server special-subject provides a WebSphere Application Server process, which runs under the server identity, to access all the CosNaming operations. The Server special-subject does not display and cannot be modified through the administrative console or other administrative tools.

Special configuration is not required to enable the server identity as specified when enabling administrative security for administrative use because the server identity is automatically mapped to the administrator role.

Users, groups, or the special subjects AllAuthenticated and Everyone can be added or removed to or from the naming roles from the WebSphere Application Server administrative console at any time. However, a server restart is required for the changes to take effect.

A best practice is to map groups or one of the special-subjects, rather than specific users, to naming roles because it is more flexible and easier to administer in the long run. By mapping a group to a naming role, adding or removing users to or from the group occurs outside of WebSphere Application Server and does not require a server restart for the change to take effect.

The CosNaming authorization policy is only enforced when administrative security is enabled. When administrative security is enabled, attempts to do CosNaming operations without the proper role assignment result in an org.omg.CORBA.NO_PERMISSION exception from the CosNaming server.

Each CosNaming function is assigned to only one role. Therefore, users who are assigned the CosNamingCreate role cannot query the name space unless they have also been assigned CosNamingRead. And in most cases a creator needs to be assigned three roles: CosNamingRead, CosNamingWrite, and CosNamingCreate. The CosNamingRead and CosNamingWrite roles assignment for the creator example are included in the CosNamingCreate role. In most of the cases, WebSphere Application Server administrators do not have to change the roles assignment for every user or group when they move to this release from a previous one.

Although the ability exists to greatly restrict access to the name space by changing the default policy, unexpected org.omg.CORBA.NO_PERMISSION exceptions can occur at runtime. Typically, Java EE applications access the name space and the identity they use is that of the user that authenticated to WebSphere Application Server when accessing the Java EE application. Unless the Java EE application provider clearly communicates the expected naming roles, use caution when changing the default naming authorization policy.

Administrative roles for business level applications:

The Java 2 Platform, Enterprise Edition (J2EE) role-based authorization concept is extended to protect the WebSphere Application Server administrative subsystem. This protection applies to those administrative roles associated with business level applications.

Deploying business level applications on a server configured to hold business level applications requires a number of administrative roles that are defined to provide degrees of authority when performing certain administrative functions from either the Web-based administrative console or the system management scripting interface. The authorization policy is only enforced when administrative security is enabled. The following table describes the system management scripting command used for business level applications and the corresponding administrative role that is required in using the command:

Table 126. Business level application - administrative roles. Business level application - administrative roles

Command	Role Required
startBLA	Cell deployer, Cell operator, BLA deployer, BLA operator, Target deployer, Target operator
stopBLA	Cell deployer, Cell operator, BLA deployer, BLA operator, Target deployer, Target operator
createEmptyBLA	Cell configurator, Cell deployer
editBLA	Cell configurator, Cell deployer, BLA deployer
viewBLA	Cell monitor, BLA monitor
listBLAs	Cell monitor, BLA monitor(s)
deleteBLA	Cell configurator, Cell deployer, BLA developer
importAsset	Cell configurator, Cell deployer
editAsset	Cell configurator, Cell deployer, Asset deployer
viewAsset	Cell monitor, Asset monitor(s)
listAssets	Cell monitor, Asset monitor
exportAsset	Cell monitor, Asset monitor
deleteAsset	Cell configurator, Cell deployer, Asset deployer
updateAsset	Cell configurator, Cell deployer, Asset deployer
addCompUnit	Cell configurator, Cell deployer, BLA deployer *(for the BLA to add the composition unit)* + Asset-deployer *(for the asset to create the composition unit from)* + Target-deployer *(for each target the composition unit is deployed to)* + Relationship-deployer *(for each relationship the composition unit depends on that will result in creating a composition unit from the dependency asset)*
editCompUnit	Cell configurator, Cell deployer, BLA deployer *(for the BLA this composition unit belongs to)* + Target deployer *(for each target that this composition unit is deployed to)*
viewCompUnit	Cell monitor, BLA monitor

Table 126. Business level application - administrative roles (continued). Business level application - administrative roles

listCompUnit	Cell monitor, BLA monitor
deleteCompUnit	Cell configurator, Cell deployer, BLA deployer *(for the BLA this composition unit belongs to)* + Target deployer *(for each target that this composition unit is deployed to)*
setCompUnitTargetAutoStart	Cell configurator, Cell deployer
listControlOps	Cell monitor, BLA monitor
getBLAStatus	Cell monitor, BLA monitor
	Where: • BLA deployer specifies the deployer role for the BLA that is being managed. • BLA monitor specifies the monitor role for the BLA that is being managed. • BLA operator specifies the operator role for the BLA that is being managed. • Asset deployer specifies the deployer role for the asset that is being managed. • Asset monitor specifies the monitor role for the asset that is being managed. • Target deployer specifies the deployer for the target that the composition unit is being deployed to. • Target operator specifies the operator role for the target that the composition unit is being deployed to.

Role-based authorization

Use authorization information to determine whether a caller has the necessary privileges to request a service.

The following figure illustrates the process that is used during authorization.

Web resource access from a web client is handled by a web collaborator. The Enterprise JavaBeans (EJB) resource access from a Java client, whether an enterprise bean or a servlet, is handled by an EJB collaborator. The EJB collaborator and the web collaborator extract the client credentials from the object request broker (ORB) current object. The client credentials are set during the authentication process as received credentials in the ORB current object. The resource and the received credentials are presented to the WSAccessManager access manager to check whether access is permitted to the client for accessing the requested resource.

The access manager module contains two main modules:

- The resource permission module helps determine the required roles for a given resource. This module uses a resource-to-roles mapping table that is built by the security runtime during application startup. To build the resource-to-role mapping table, the security runtime reads the deployment descriptor of the enterprise beans or the Web module (`ejb-jar.xml` file or `web.xml` file)
- The authorization table module consults a role-to-user or group table to determine whether a client is granted one of the required roles. The role-to-user or group mapping table, also known as the *authorization table*, is created by the security runtime during application startup.

  To build the authorization table, the security run time reads the application binding file, the `ibm-application-bnd.xmi` file, or the `ibm-application-bnd.xml` file, as appropriate.

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named ibm-*-ext.xmi or ibm-*-bnd.xmi where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

Use authorization information to determine whether a caller has the necessary privilege to request a service. You can store authorization information many ways. For example, with each resource, you can store an access-control list, which contains a list of users and user privileges. Another way to store the information is to associate a list of resources and the corresponding privileges with each user. This list is called a *capability list*.

WebSphere Application Server uses the Java 2 Platform, Enterprise Edition (J2EE) authorization model. In this model, authorization information is organized as follows:

During the assembly of an application, permission to invoke methods is granted to one or more roles. A role is a set of permissions; for example, in a banking application, roles can include teller, supervisor, clerk, and other industry-related positions. The teller role is associated with permissions to run methods that are related to managing the money in an account, such as the withdraw and deposit methods. The teller role is not granted permission to close accounts; this permission is given to the supervisor role. The application assembler defines a list of method permissions for each role. This list is stored in the deployment descriptor for the application.

Three *special subjects* are not defined by the J2EE model: AllAuthenticatedUsers, AllAuthenticatedInTrustedRealms, and Everyone. A special subject is a product-defined entity that is defined outside of the user registry. This entity is used to generically represent a class of users or groups in the registry.

- The AllAuthenticatedUsers subject permits all authenticated users to access protected methods. As long as the user can authenticate successfully, the user is permitted access to the protected resource.
- The AllAuthenticatedInTrustedRealms subject permits all authenticated foreign users (those that are bound to other realms) to access protected methods. As long as the user can authenticate successfully, the user is permitted access to the protected resource.
- The Everyone subject permits unrestricted access to a protected resource. Users do not have to authenticate to get access; this special subject provides access to protected methods as if the resources are unprotected.

During the deployment of an application, real users or groups of users are assigned to the roles. When a user is assigned to a role, the user gets all the method permissions that are granted to that role.

The application deployer does not need to understand the individual methods. By assigning roles to methods, the application assembler simplifies the job of the application deployer. Instead of working with a set of methods, the deployer works with the roles, which represent semantic groupings of the methods.

Users can be assigned to more than one role; the permissions that are granted to the user are the union of the permissions granted to each role. Additionally, if the authentication mechanism supports the grouping of users, these groups can be assigned to roles. Assigning a group to a role has the same effect as assigning each individual user to the role.

A best practice during deployment is to assign groups instead of individual users to roles for the following reasons:
- Improves performance during the authorization check. Typically far fewer groups exist than users.
- Provides greater flexibility, by using group membership to control resource access.
- Supports the addition and deletion of users from groups outside of the product environment. This action is preferred to adding and removing them to WebSphere Application Server roles. Stop and restart the enterprise application for these changes to take effect. This action can be very disruptive in a production environment.

At runtime, WebSphere Application Server authorizes incoming requests based on the user's identification information and the mapping of the user to roles. If the user belongs to any role that has permission to run a method, the request is authorized. If the user does not belong to any role that has permission, the request is denied.

The J2EE approach represents a declarative approach to authorization, but it also recognizes that you cannot deal with all situations declaratively. For these situations, methods are provided for determining user and role information programmatically. For enterprise beans, the following two methods are supported by WebSphere Application Server:
- **getCallerPrincipal**: This method retrieves the user identification information.
- **isCallerInRole**: This method checks the user identification information against a specific role.

For servlets, the following methods are supported by WebSphere Application Server:
- getRemoteUser
- isUserInRole
- getUserPrincipal

These methods correspond in purpose to the enterprise bean methods.

For more information on the J2EE security authorization model, see the following website: http://java.sun.com

Administrative roles

The Java Platform, Enterprise Edition (Java EE) role-based authorization concept is extended to protect the WebSphere Application Server administrative subsystem.

A number of administrative roles are defined to provide degrees of authority that are needed to perform certain administrative functions from either the Web-based administrative console or the system management scripting interface. The authorization policy is only enforced when administrative security is enabled. The following table describes the administrative roles:

Table 127. Administrative roles. Administrative roles

Role	Description
Monitor	An individual or group that uses the monitor role has the least amount of privileges. A monitor can complete the following tasks: • View the WebSphere Application Server configuration. • View the current state of the Application Server.

Table 127. Administrative roles (continued). Administrative roles

Configurator	An individual or group that uses the configurator role has the monitor privilege plus the ability to change the WebSphere Application Server configuration. The configurator can perform all the daily configuration tasks. For example, a configurator can complete the following tasks: • Create a resource. • Map an application server. • Install and uninstall an application. • Deploy an application. • Assign users and groups-to-role mapping for applications. • Set up Java 2 security permissions for applications. • Customize the Common Secure Interoperability Version 2 (CSIv2), Security Authentication Service (SAS), and Secure Sockets Layer (SSL) configurations. **Important:** SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.
Operator	An individual or group that uses the operator role has monitor privileges plus ability to change the runtime state. For example, an operator can complete the following tasks: • Stop and start the server. • Monitor the server status in the administrative console.
Administrator	An individual or group that uses the administrator role has the operator and configurator privileges, plus additional privileges that are granted solely to the administrator role. For example, an administrator can complete the following tasks: • Modify the server user ID and password. • Configure authentication and authorization mechanisms. • Enable or disable administrative security. • Enable or disable Java 2 security. • Change the Lightweight Third Party Authentication (LTPA) password and generate keys. • Create, update, or delete users in the federated repositories configuration. • Create, update, or delete groups in the federated repositories configuration. **Important:** An administrator cannot map users and groups to the administrator roles without also having the adminsecuritymanager role.
iscadmins	This role is only available for administrative console users, not for wsadmin users. Users who are granted this role have administrator privileges for managing users and groups in the federated repositories. For example, a user of the iscadmins role can complete the following tasks: • Create, update, or delete users in the federated repositories configuration. • Create, update, or delete groups in the federated repositories configuration.
Deployer	Users granted this role can complete both configuration actions and runtime operations on applications. See the “Deployer role” on page 1625 section for more details.
Admin Security Manager	You can assign users and groups to the Admin Security Manager role on the cell level through wsadmin scripts and the administrative console. Using the Admin Security Manager role, you can assign users and groups to the administrative user roles and administrative group roles. However, an administrator cannot assign users and groups to the administrative user roles and administrative group roles including the Admin Security Manager role. See the “Admin Security Manager role” on page 1626 section for more details.

Table 127. Administrative roles (continued). Administrative roles

Auditor	Users granted this role can view and modify the configuration settings for the security auditing subsystem. For example, a user with the auditor role can complete the following tasks: • Enable and disable the security auditing subsystem. • Select the event factory implementation to be used with the event factory plug-in point. • Select and configure the service provide, or emitter. or both to be used with the service provider plug-in point. • Set the audit policy that describes the behavior of the application server in the event of an error with the security auditing subsystem. • Define which security events are to be audited. The auditor role includes the monitor role. This allows the auditor to view but not change the rest of the security configuration. See the "Auditor role" on page 1626 section for more details.

The server ID that is specified and the administrative ID, if specified, when enabling administrative security is automatically mapped to the administrator role.

Users and groups can be added or removed from administrative roles using the WebSphere Application Server administrative console by a user given the appropriate authority. The Primary administrative user name must be used to log on to the administrative console to change the administrative user and group roles other than the auditor role. Only a user with the auditor role can change the auditor user and group roles. When security auditing is initially enabled, the Primary administrative user is also given the auditor role, and can manage all of the administrative user and group roles including the those in the auditor role. A best practice is to map a group or groups, rather than specific users, to administrative roles because it is more flexible and easier to administer.

In addition to mapping user or groups, a special-subject can also be mapped to the administrative roles. A special-subject subject is a generalization of a particular class of users. The AllAuthenticated special subject means that the access check of the administrative role ensures that the user making the request is at least authenticated. The Everyone special subject means that anyone, authenticated or not, can perform the action, as if security was not enabled.

Deployer role

A user that is granted a deployer role can complete all of the configuration and runtime operations on an application. A deployer role can be subsets of both configurator and operator roles. However, a user granted a deployer role cannot configure or operate any other resources, such as a server, node.

When fine-grained administrative security is used, only a user granted a deployer role to an application can configure and operate that application.

Cell-level configurators can configure applications. Cell-level operators can also operate (start and stop) applications. However, a user granted a deployer role at cell level can also complete configuration and operation on all applications.

Table 128. Deployer role capabilities.

This table lists the deployer role capabilities when fine-grained administrative security is used.

Operation	Required Roles (Any one)
Install application	Cell-configurator, target-deployer
Uninstall application	Cell-configurator, application-deployer, target-deployer
List application	Cell-monitor, application-monitor

Table 128. Deployer role capabilities (continued).

This table lists the deployer role capabilities when fine-grained administrative security is used.

Operation	Required Roles (Any one)
Edit, update and redeploy application	Cell-configurator, application-deployer
Export application	Cell-monitor, application-monitor
Start or stop application	Cell-operator, application-deployer

Where:

Cell-configurator
Specifies the configurator role at cell level.

Application-deployer
Specifies the deployer role for the application that is being managed.

Target-deployer
Specifies the deployer role for all servers or clusters for which an application is targeted. If you have a target-deployer role, you can install a new application on the target. However, to edit or update the installed application, you must be included in the authorization group of the installed application-deployer.

The target-deployer cannot explicitly start or stop a new application. However, when a target-deployer starts a server on a target, all of the applications that have their auto-start attribute set to `yes` are started when the server starts.

It is recommended that the application-deployer set this attribute to `true` if the application-deployer does not want the application to be started by the target-deployer.

Admin Security Manager role

The Admin Security Manager role separates administrative security administration from other application administration.

By default, serverId and adminID, if specified, are assigned to this role in the cell level authorization table. This role implies a monitor role. However, an administrator role does not imply the Admin Security Manager role.

When fine-grained admin security is used, only a user granted this role at cell level can manage administrative authorization groups. However, a user granted this role for each administrative authorization group can map users to administrative roles for those groups. The following list summarizes the capabilities of the Admin Security Manager role at different levels, such as the cell and administrative authorization group levels.

Table 129. Admin Security Manager role capabilities.

This table lists Admin Security Manager role capabilities.

Action	Role
Map users to administrative roles for cell level	Only the Admin Security Manager of the cell
Map users to administrative roles for an authorization group	Only the Admin Security Manager of that authorization group or the Admin Security Manager of the cell
Manage authorization groups, create, delete, add resource to an authorization group, or remove resource from an authorization group or list	Only the Admin Security Manager of the cell

Auditor role

The auditor role separates security auditing administration from administrative security and other application administration.

The auditor role was added to allow distinct separation of the authority of an auditor from the authority of the administrator. The auditor role can be granted to administrators to combine their authority. When security is first enabled, the auditor role is assigned to the primary administrator. If in your situation the separation of authority is required, administrators can remove the auditor role from themselves and assign the auditor role to other users.

A fine grained security for the auditor role is not implemented, which results in the auditor role requiring the monitor role. This process allows the auditor to read but not modify the panels managed by the administrator. The auditor has full authority to read and modify the panels associated with the security auditing subsystem. The administrator will have the monitor role for those panels, however, the administrator cannot modify those panels.

Authorization providers

WebSphere Application Server supports authorization that is based on the Java Authorization Contract for Containers (JACC) specification in addition to the default authorization.

JACC is a specification introduced in Java Platform, Enterprise Edition (Java EE)1.4. It enables third-party security providers to manage authorization in the application server.

Note: In WebSphere Application Server version 7.0, Java Authorization Contract for Containers (JACC) specification 1.4 was applied. In JACC specification 1.4, we support Java EE5 that includes Servlet 2.5 and EJB 3. The biggest functional change with the introduction of JACC specification 1.4 is the inclusion of annotations for propagating security policy information.

When security is enabled in WebSphere Application Server, the default authorization is used unless a JACC provider is specified. The default authorization does not require special setup, and the default authorization engine makes all of the authorization decisions. However, if a JACC provider is configured and set up for WebSphere Application Server to use, all of the enterprise beans and web authorization decisions are delegated to the JACC provider.

WebSphere Application Server supports security for Java EE applications and also for its administrative components. Java EE applications, such as Web and Enterprise JavaBeans (EJB) components are protected and authorized per the Java EE specification. The administrative components are internal to WebSphere Application Server and are protected by the role-based authorizer. The administrative components include the administrative console, MBeans, and other components such as naming and security. For more information on administrative security, see “Role-based authorization” on page 1621.

When a JACC provider is used for authorization in WebSphere Application Server, all of the Java EE application-based authorization decisions are delegated to the provider per the JACC specification. However, all administrative security authorization decisions are made by the WebSphere Application Server default authorization engine. The JACC provider is not called to make the authorization decisions for administrative security.

When a protected Java EE resource is accessed, the authorization decision to give access to the principal is the same whether using the default authorization engine or a JACC provider. Both of the authorization models satisfy the J2EE specification, and function the same. Choose a JACC provider only when you want to work with an external security provider such as Tivoli Access Manager. In this instance, the security provider must support the JACC specification and be set up to work with WebSphere Application Server. Setting up and configuring a JACC provider requires additional configuration steps, depending on the provider. Unless you have an external security provider that you can use with WebSphere Application Server, use the default authorization.

JACC support in WebSphere Application Server:

WebSphere Application Server supports the Java Authorization Contract for Containers (JACC) specification, which enables third-party security providers to handle the Java Platform, Enterprise Edition (Java EE) authorization.

The JACC specification requires that both the containers in the application server and the provider satisfy some requirements. Specifically, the containers are required to propagate the security policy information to the provider during the application deployment and to call the provider for all authorization decisions. The providers are required to store the policy information in their repository during application deployment. The providers then use this information to make authorization decisions when called by the container.

JACC access decisions

When security is enabled and an enterprise bean or web resource is accessed, the Enterprise JavaBeans (EJB) container or web container calls the security runtime to make an authorization decision on whether to permit access. When using an external provider, the access decision is delegated to that provider.

According to the Java Authorization Contract for Containers (JACC) specification, the appropriate permission object is created, the appropriate policy context handlers are registered, and the appropriate policy context identifier (contextID) is set. A call is made to the java.security.Policy object method that is implemented by the provider to make the access decision.

The following sections describe how the provider is called for both the enterprise bean and the web resources.

Access decisions for enterprise beans

When security is enabled, and an EJB method is accessed, the EJB container delegates the authorization check to the security runtime. If JACC is enabled, the security runtime uses the following process to perform the authorization check:

1. Creates the EJBMethodPermission object using the bean name, method name, interface name, and the method signature.
2. Creates the context ID and sets it on the thread by using the PolicyContext.setContextID(contextID) method.
3. Registers the required policy context handlers, including the Subject policy context handler.
4. Creates the ProtectionDomain object with principal in the Subject. If no principal exists, null is passed for the principal name.
5. The access decision is delegated to the JACC provider by calling the implies method of the Policy object, which is implemented by the provider. The EJBMethodPermission and the ProtectionDomain objects are passed to this method.
6. The isCallerInRole access check also follows the same process, except that an EJBRoleRefPermission object is created instead of an EJBMethodPermission object.

Access decisions for web resources

When security is enabled and configured to use a JACC provider, and when a web resource such as a servlet or a JavaServer Pages (JSP) file is accessed, the security runtime delegates the authorization decision to the JACC provider by using the following process:

1. A WebResourcePermission object is created to see if the URI is cleared. If the provider honors the Everyone subject it is also selected here.
   a. The WebResourcePermission object is constructed with the urlPattern and the HTTP method accessed.
   b. A ProtectionDomain object with a null principal name is created.

c. The JACC provider Policy.implies method is called with the permission and the protection domain. If the URI access is cleared or given access to the Everyone subject, the provider permits access (return `true`) in the implies method. Access is then granted without further checks.

2. If access is not granted in the previous step, a WebUserDataPermission object is created and used to see if the Uniform Resource Identifier (URI) is precluded, excluded or must be redirected using the HTTPS protocol.
   a. The WebUserDataPermission object is constructed with the urlPattern accessed, the HTTP method invoked, and the transport type of the request. If the request is over HTTPS, the transport type is set to CONFIDENTIAL; otherwise, null is passed.
   b. A ProtectionDomain object with a null principal name is created.
   c. The JACC provider Policy.implies method is called with the permission and the protection domain. If the request is using the HTTPS protocol and the implies method returns `false`, the HTTP 403 error is returned to imply excluded and precluded permission. In this case no further checks are performed. If the request is not using the HTTPS protocol, and the implies returns `false`, the request is redirected over HTTPS.
3. The security runtime attempts to authenticate the user. If the authentication information already exists (for example, LTPA token), it is used. Otherwise, the user's credentials must be entered.
4. After the user credentials are validated, a final authorization check is performed to see if the user is granted access privileges to the URI.
   a. As in Step 1, the WebResourcePermission object is created. The ProtectionDomain object now contains the Principal that is attempting to access the URI. The Subject policy context handler also contains the user's information, which can be used for the access check.
   b. The provider implies method is called using the Permission object and the ProtectionDomain object created previously. If the user is granted permission to access the resource, the implies method returns `true`. If the user is not granted access, the implies method returns `false`.

Even if the order listed previously is changed later (for example, to improve performance) the end result is the same. For example, if the resource is precluded or excluded, the end result is that the resource cannot be accessed.

For more information on these access permissions, see the JSR-000115 Java Authorization Contract for Containers (Final Release).

Using information from the Subject for access decision

If the provider relies on the WebSphere Application Server generated Subject for access decision, the provider can query the public credentials in the Subject to obtain the WSCredential credential. The WSCredential API is used to obtain information about the user, including the name and the groups that the user belongs to. This information is used to make the access decision.

If the provider adds the required information to the Subject, WebSphere Application Server can use the information to make the access decision. The provider might add the information by using the Trust Association Interface feature or by plugging login modules into the Application Server.

The security attribute propagation section contains additional documentation on how to add the WebSphere Application Server required information to the Subject. For more information, see “Propagating security attributes among application servers” on page 1549.

Dynamic module updates in JACC

WebSphere Application Server supports dynamic updates to web modules under certain conditions. If a web module is updated, deleted or added to an application, only that module is stopped and started as appropriate. The other existing modules in the application are not impacted, and the application itself is not stopped and then restarted.

When using the default authorization engine, any security policies are modified in the web modules and the application is stopped and then restarted. When using the Java Authorization Contract for Containers (JACC) based authorization, the behavior depends on the functionality that a provider supports. If a provider can handle dynamic changes to the web modules, then only the web modules are impacted. Otherwise, the entire application is stopped and restarted for the new changes in the web modules to take effect.

A provider can indicate if it supports the dynamic updates by configuring the **Supports dynamic module updates** option in the JACC configuration model (see “Authorizing access to Java EE resources using Tivoli Access Manager” on page 1644 for more information). This option can be enabled or disabled using the administrative console or by scripting. It is expected that most providers store the policy information in their external repository, which makes it possible for them to support these dynamic updates. This option should be enabled by default for most providers.

When the **Supports dynamic module updates** option is enabled, if a web module that contains security roles is dynamically added, modified, or deleted, only the specific web modules are impacted and restarted. If the option is disabled, the entire application is restarted. When dynamic updates are performed, the security policy information of the modules impacted are propagated to the provider. For more information about security policy propagation, see “JACC policy propagation” on page 1632.

Initialization of the JACC provider

If a Java Authorization Contract for Containers (JACC) provider requires initialization during server startup, for example, to enable the client code to communicate to the server code, the provider can implement the com.ibm.wsspi.security.authorization.InitializeJACCProvider interface. See “Interfaces that support JACC” on page 1662 for more information.

When this interface is implemented, it is called during server startup. Any custom properties in the JACC configuration model are propagated to the initialize method of this implementation. The custom properties can be entered using either the administrative console or by scripting.

During server shutdown, the cleanup method is called for any clean-up work that a provider requires. Implementation of this interface is strictly optional, and is used only if the provider requires initialization during server startup.

Mixed node environment and JACC

Authorization using Java Authorization Contract for Containers (JACC) is a new feature in WebSphere Application Server Version 6.0.*x*. Also, the JACC configuration is set up at the cell level and is applicable for all the nodes and servers in that cell.

If you are planning to use the JACC-based authorization, the cell must contain Version 6.0.*x* and later nodes only. This restriction implies that a mixed node environment containing a set of Version 5.*x* nodes in a Version 6.0.*x* or later cell is not supported.

JACC providers:

The Java Authorization Contract for Containers (JACC) is a specification that was first introduced in Java Platform, Enterprise Edition (Java EE) Version 1.4 through the Java Specifications Request (JSR) 115 process. JACC specification 1.4 is included for WebSphere Application Server version 7.0 for Java EE 5 support.. This specification defines a contract between Java EE 5 containers and authorization providers.

The contract enables third-party authorization providers to plug into Java EE 5 application servers, such as WebSphere Application Server, to make the authorization decisions when a Java EE 5 resource is accessed. The access decisions are made through the standard java.security.Policy object.

To plug in to WebSphere Application Server, the third-party JACC provider must implement the policy class, policy configuration factory class, and policy configuration interface, which are all required by the JACC specification.

The JACC specification does not specify how to handle the authorization table information between the container and the provider. It is the responsibility of the provider to provide some management facilities to handle this information. The container is not required to provide the authorization table information in the binding file to the provider.

WebSphere Application Server provides the RoleConfigurationFactory and the RoleConfiguration role configuration interfaces to help the provider obtain information from the binding file, as well as an initialization interface (InitializeJACCProvider). The implementation of these interfaces is optional. See "Interfaces that support JACC" on page 1662 for more information about these interfaces.

Tivoli Access Manager as the default JACC provider for WebSphere Application Server

The JACC provider in WebSphere Application Server is implemented by both the client and the server pieces of the Tivoli Access Manager. The client piece of Tivoli Access Manager is embedded in WebSphere Application Server. The server piece is located on a separate installable CD that is shipped as part of the WebSphere Application Server, Network Deployment (ND) package.

The JACC provider is not the default authorization. You must configure WebSphere Application Server to use the JACC provider.

JACC policy context handlers:

WebSphere Application Server supports all of the policy context handlers that are required by the Java Authorization Contract for Containers (JACC) specification. However, due to performance impacts, the Enterprise JavaBeans (EJB) arguments policy context handler is not activated unless it is specifically required by the provider. Performance impacts result if objects must be created for each argument of each EJB method.

If the provider supports and requires this context handler, select the **Requires the EJB arguments policy context handler for access decisions** check box in the External JACC provider link under the Authorization providers panel or by using scripting. Any changes to this option are effective after the servers are restarted. By default this option is disabled. Disable this option when using Tivoli Access Manager as the JACC provider, because the argument values are not required for access decisions.

JACC policy context identifiers (ContextID) format:

A policy context identifier is defined as a unique string that represents a policy context. A policy context contains all of the security policy statements as defined by the Java Contract for Containers (JACC) specification that affect access to the resources in a web or Enterprise JavaBeans (EJB) module.

During policy propagation to the JACC provider, a PolicyConfiguration object is created for each policy context. The object is populated with the policy statements, represented by the JACC permission objects that correspond to the context. The object is propagated to the JACC provider using the JACC specification APIs.

Note: The following information is include for planning purposes and is only applicable if you intend to federate in the future.

WebSphere Application Server makes the contextID unique by using the `href:cellName/appName/moduleName` string as the contextID format for the modules. The `href` part of the string indicates that a hierarchical name is passed as the context ID. The `cellName` represents the name of the deployment manager cell or the base cell where the application is installed.

The appName part of the string in the context ID represents the application name containing the module. The moduleName refers to the name of the module.

As an example, the context ID for the module Increment.jar file in an application named DefaultApplication that is installed in cell1 is the href:cell1/DefaultApplication/Increment.jar file.

JACC policy propagation:

When an application is installed or deployed in WebSphere Application Server, the security policy information in the application is propagated to the provider when the configuration is saved. The context ID for the application is saved in its application.xml file, that is used for propagating the policy to the Java Authorization Contract for Containers (JACC) provider, and also for access decisions for Java Platform, Enterprise Edition (Java EE) resources.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log , SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

When an application is uninstalled, the security policy information in the application is removed from the provider when the configuration is saved.

If the provider implemented the RoleConfiguration interface, the security policy information that is propagated to the policy provider also contains the authorization table information. See “Interfaces that support JACC” on page 1662 for more information about this interface.

If an application does not contain security policy information, the PolicyConfiguration (and the RoleConfiguration, if implemented) objects do not contain any information. The existence of empty PolicyConfiguration and RoleConfiguration objects indicates that security policy information for the module does not exist.

After an application is installed, it can be updated without being uninstalled and reinstalled. For example, a new module can be added to an existing application, or an existing module can be modified. In this instance, the information in the impacted modules is propagated to the provider by default. A module is impacted when the deployment descriptor of the module or annotations within the module are changed as part of the update. If the provider supports the RoleConfiguration interfaces, the entire authorization table for that application is propagated to the provider.

If the security information is not propagated to the provider during application updates, you can set the com.ibm.websphere.security.jacc.propagateonappupdate Java virtual machine (JVM) property to false in the deployment manager, in a Network Deployment environment, or the unmanaged base application server. If this property is set to false, any updates to an existing application in the server are not propagated to the provider. You also can set this property on a per-application basis using the custom properties of an application. The wsadmin tool can be used to set the custom property of an application. If this property is set at the application level, any updates to that application are not propagated to the provider. If the update to an application is a full update, for example, a new application enterprise archive (EAR) file is used to replace the existing one, and the provider is refreshed with the entire application security policy information.

As mentioned earlier, the security policy information is propagated to the JACC provider during the save operation. The SystemOut.log file indicates the success or failure of the propagation to the provider. Check the log file after the installation to ensure that the propagation had no problems. If the propagation had any problems, access to the application fails when Tivoli Access Manager is used as the JACC provider.

If the security policy information for the application is successfully propagated to the provider, the audit statements with the message key SECJ0415I appear. However, if there was a problem propagating the security policy information to the provider (for example: network problems, JACC provider is not available), the `SystemOut.log` files contain the error message with the message keys SECJ0396E during install or SECJ0398E during modification. The installation of the application is not stopped due to a failure to propagate the security policy to the JACC provider. Also, in the case of failure, no exception or error messages appear during the save operation. When the problem causing this failure is fixed, run the **propagatePolicyToJaccProvider** tool to propagate the security policy information to the provider without reinstalling the application. For more information, see Propagating security policy of installed applications to a JACC provider using wsadmin scripting.

JACC registration of the provider implementation classes:

The JACC specification states that providers can plug in their provider using the javax.security.jacc.policy.provider and the javax.security.jacc.PolicyConfigurationFactory.provider system properties.

The javax.security.jacc.policy.provider property is used to set the policy object of the provider, while the javax.security.jacc.PolicyConfigurationFactory.provider property is used to set the provider PolicyConfigurationFactory implementation.

Although both system properties are supported in WebSphere Application Server, it is highly recommended that you use the configuration model that is provided. You can set these values using either the JACC configuration panel (see “Authorizing access to Java EE resources using Tivoli Access Manager” on page 1644 for more information) or by using wsadmin scripting. One of the advantages of using the configuration model instead of the system properties is that the information is entered in one place at the cell level, and is propagated to all nodes during synchronization. Also, as part of the configuration model, additional properties can be entered, as described in the JACC configuration panel.

Role-based security with embedded Tivoli Access Manager:

The Java Platform, Enterprise Edition (Java EE) role-based authorization model uses the concepts of roles and resources. An example is provided here.

Table 130. Roles.

This table is an example of role-based security with embedded Tivoli Access Manager.

	Methods		
Roles	**getBalance**	**deposit**	**closeAccount**
Teller	granted	granted	
Cashier	granted		
Supervisor			granted

In the example of the banking application that is conceptualized in the previous table, three roles are defined: teller, cashier, and supervisor. Permission to perform the getBalance, deposit, and closeAccount application methods are mapped to these roles. From the example, you can see that users assigned the role, Supervisor, can run the closeAccount method, whereas the other two roles are unable to run this method.

The term, principal, within WebSphere Application Sever security refers to a person or a process that performs activities. *Groups* are logical collections of principals that are configured in WebSphere Application Server to promote the ease of applying security. Roles can be mapped to principals, groups, or both.

Table 131. Roles methods. The entry that is invoked in the following table indicates that the principal or group can invoke any methods that are granted to that role.

	Roles		
Principal/Group	**Teller**	**Cashier**	**Supervisor**
TellerGroup	Invoke		
CashierGroup		Invoke	
SupervisorGroup			
Frank: A principal who is not a member of any of the previous groups		Invoke	Invoke

In the previous example, the principal Frank, can invoke the getBalance and the closeAccount methods, but cannot invoke the deposit method because this method is not granted either the Cashier or the Supervisor role.

At the time of application deployment, the Java Authorization Contract for Container (JACC) provider of Tivoli Access Manager populates the Tivoli Access Manager-protected object space with any security policy information that is contained in the application deployment descriptor and or annotations. This security information is used to determine access whenever the WebSphere Application Server resource is requested.

By default, the Tivoli Access Manager access check is performed using the role name, the cell name, the application name, and the module name.

Tivoli Access Manager access control lists (ACLs) determine which application roles are assigned to a principal. ACLs are attached to the applications in the Tivoli Access Manager-protected object space at the time of application deployment.

Principal-to-role mappings are managed from the WebSphere Application Server administrative console and are never modified using Tivoli Access Manager. Direct updates to ACLs are performed for administrative security users only.

The following sequence of events occur:

1. During application deployment, policy information is sent to the JACC provider of Tivoli Access Manager . This policy information contains permission-to-role mappings and role-to-principal and role-to-group mapping information.
2. The JACC provider of Tivoli Access Manager converts the information into the required format, and passes this information to the Tivoli Access Manager policy server.
3. The policy server adds entries to the Tivoli Access Manager-protected object space to represent the roles that are defined for the application and the permission-to-role mappings. A permission is represented as a Tivoli Access Manager-protected object and the role that is granted to this object is attached as an extended attribute.

Tivoli Access Manager integration as the JACC provider:

Tivoli Access Manager uses the Java Authorization Contract for Container (JACC) model in WebSphere Application Server to perform access checks.

Tivoli Access Manager consists of the following components:

- Run time
- Client configuration
- Authorization table support
- Access check
- Authentication using the PDLoginModule module

Tivoli Access Manager run-time changes that are used to support JACC

For the run-time changes, Tivoli Access Manager implements the PolicyConfigurationFactory and the PolicyConfiguration interfaces, as required by JACC. During the application installation, the security policy information in the deployment descriptor and the authorization table information in the binding files are propagated to the Tivoli provider using these interfaces. The Tivoli provider stores the policy and the authorization table information in the Tivoli Access Manager policy server by calling the respective Tivoli Access Manager application programming interfaces (API).

Tivoli Access Manager also implements the RoleConfigurationFactory and the RoleConfiguration interfaces. These interfaces are used to ensure that the authorization table information is passed to the provider with the policy information. See “Interfaces that support JACC” on page 1662 for more information about these interfaces.

Tivoli Access Manager client configuration

To configure the Tivoli Access Manager client, you can use either the administrative console or wsadmin scripting. You can access the administrative console panels for the Tivoli Access Manager client configuration by clicking **Security > Global security > External authorization providers**. Under Related Items, click **External JACC provider**. The Tivoli client must be set up to use the Tivoli Access Manager JACC Provider.

For more information about how to configure the Tivoli Access Manager client, see “Tivoli Access Manager JACC provider configuration” on page 1651.

Authorization table support

Tivoli Access Manager uses the RoleConfiguration interface to ensure that the authorization table information is passed to the Tivoli Access Manager provider when the application is installed or deployed. When an application is deployed or edited, the set of users and groups for the user or group-to-role mapping are obtained from the Tivoli Access Manager server, which shares the same Lightweight Directory Access Protocol (LDAP) server as WebSphere Application Server. This sharing is accomplished by plugging into the application management users or groups-to-role administrative console panels. The management APIs are called to obtain users and groups rather than relying on the WebSphere Application Server-configured LDAP registry.

Access check

When WebSphere Application Server is configured to use the JACC provider for Tivoli Access Manager , it passes the information to Tivoli Access Manager to make the access decision. The Tivoli Access Manager policy implementation queries the local replica of the access control list (ACL) database for the access decision.

Authentication using the PDLoginModule module

The custom login module in WebSphere Application Server can do the authentication. This login module is plugged in before the WebSphere Application Server-provided login modules. The custom login modules can provide information that can be stored in the Subject. If the required information is stored, no additional registry calls are made to obtain that information.

As part of the JACC integration, the Tivoli Access Manager-provided PDLoginModule module is also used to plug into WebSphere Application Server for Lightweight Third Party Authentication (LTPA), Kerberos (KRB5) and Simple WebSphere Authentication Mechanism (SWAM) authentication. The PDLoginModule module is modified to authenticate with the user ID or password. The module is also used to fill in the

required attributes in the Subject so that no registry calls are made by the login modules in WebSphere Application Server. The information that is placed in the Subject is available for the Tivoli Access Manager policy object to use for access checking.

Note: SWAM is deprecated in WebSphere Application Server Version 8.0 and will be removed in a future release.

Note: When using Kerberos authentication mechanism and Tivoli Access Manager, TAM loginModule creates the PDPrincipal without first going through the Tivoli Access Manager authentication process. Also when using Kerberos authentication mechanism and Tivoli Access Manager, the Tivoli Access Manager policy is not enforced starting in WebSphere Application Server Version 7.0.

Tivoli Access Manager security for WebSphere Application Server:

WebSphere Application Server provides embedded IBM Tivoli Access Manager client technology to secure your WebSphere Application Server-managed resources.

The benefits of using Tivoli Access Manager that are described here are only applicable when Tivoli Access Manager client code is used with the Tivoli Access Manager server:

- Robust container-based authorization
- Centralized policy management
- Management of common identities, user profiles, and authorization mechanisms
- Single-point security management for Java Platform, Enterprise Edition (Java EE) compliant and non-compliant Java EE resources using the administrative console for Tivoli Access Manager Web Portal Manager
- No requirements for coding or deployment changes to applications
- Easy management of users, groups, and roles using the WebSphere Application Server administrative console

WebSphere Application Server supports the Java Authorization Contract for Containers (JACC) specification. JACC details the contract requirements for Java EE containers and authorization providers. With this contract, authorization providers can perform the access decisions for resources in Java EE application servers such as WebSphere Application Server. The Tivoli Access Manager security utility that is embedded within WebSphere Application Server is JACC-compliant and is used to:

- Add security policy information when applications are deployed
- Authorize access to WebSphere Application Server-secured resources.

When applications are deployed, the embedded Tivoli Access Manager client takes any policy and or user and role information that is stored within the application deployment descriptor or using annotations and stores it within the Tivoli Access Manager Policy Server.

The Tivoli Access Manager JACC provider is also called when a user requests access to a resource that is managed by WebSphere Application Server.

Embedded Tivoli Access Manager client architecture

The previous figure illustrates the following sequence of events:

1. Users that access protected resources are authenticated using the Tivoli Access Manager login module that is configured for use when the embedded Tivoli Access Manager client is enabled.
2. The WebSphere Application Server container uses information from the Java EE application deployment descriptor and annotations to determine the required role membership.

3. WebSphere Application Server uses the embedded Tivoli Access Manager client to request an authorization decision from the Tivoli Access Manager authorization server. Additional context information, when present, is also passed to the authorization server. This context information is comprised of the cell name, Java EE application name, and Java EE module name. If the Tivoli Access Manager policy database has policies that are specified for any of the context information, the authorization server uses this information to make the authorization decision.
4. The authorization server consults the permissions that are defined for the specified user within the Tivoli Access Manager-protected object space. The protected object space is part of the policy database.
5. The Tivoli Access Manager authorization server returns the access decision to the embedded Tivoli Access Manager client.
6. WebSphere Application Server either grants or denies access to the protected method or resource, based on the decision that is returned from the Tivoli Access Manager authorization server.

At its core, Tivoli Access Manager provides an authentication and authorization framework. You can learn more about Tivoli Access Manager, including the information that is necessary to make deployment decisions, by reviewing the product documentation. The following guides are available in the IBM Tivoli Access Manager for e-business Information Center:

- *IBM Tivoli Access Manager for e-business Installation Guide*

  This guide describes how to plan, install, and configure a Tivoli Access Manager secure domain. Using a series of easy installation scripts, you can quickly deploy a fully functional secure domain. These scripts are very useful when prototyping the deployment of a secure domain.

  To access this guide in the IBM Tivoli Access Manager for e-business information center, click **Access Manager for e-business > Installation and upgrade information > Installation Guide**.
- *IBM Tivoli Access Manager for e-business Administration Guide*

  This document presents an overview of the Tivoli Access Manager security model for managing protected resources. This guide describes how to configure the Tivoli Access Manager servers that make access control decisions. In addition, detailed instructions describe how to perform important tasks, such as declaring security policies, defining protected object spaces, and administering user and group profiles.

  To access this guide in the IBM Tivoli Access Manager for e-business information center, click **Access Manager for e-business >Administration Information > Administration Guide**.

Tivoli Access Manager provides centralized administration of multiple servers.

The previous figure is an example architecture showing WebSphere Application Servers secured by Tivoli Access Manager.

The participating WebSphere Application Servers use a local replica of the Tivoli Access Manager policy database to make authorization decisions for incoming requests. The local policy databases are replicas of the master policy database. The master policy database is installed as part of the Tivoli Access Manager installation. Having policy database replicas on each participating WebSphere Application Server node optimizes performance when making authorization decisions and provides failover capability.

Although the authorization server can also be installed on the same system as WebSphere Application Server, this configuration is not illustrated in the diagram.

All instances of Tivoli Access Manager and WebSphere Application Server in the example architecture share the Lightweight Directory Access Protocol (LDAP) user registry on Machine E.

The LDAP registries that are supported by WebSphere Application Server are also supported by Tivoli Access Manager.

It is possible to have separate WebSphere Application Server profiles on the same host that is configured for different Tivoli Access Manager servers. Such an architecture requires that the profiles are configured for separate Java SE Runtime Environments (JRE 6) and therefore you need multiple JREs installed on the same host.

Note: Even though all WebSphere Application Server profiles on the same host share a single JRE 6, you can configure separate WebSphere Application Server profiles on the same host for different Tivoli Access Manager servers.

Security annotations:

Annotations are a powerful programming mechanism resulting from the JSR-175 recommendation. An annotation is a standard way to include supported security behaviors while allowing, the source code and configuration files to be generated automatically.

In Java Platform, Enterprise Edition (Java EE) 5 and above, The security roles and policies can be defined using annotations as well as within the deployment descriptor. During the installation of the application, the security policies and roles defined using annotations are merged with the security policies and roles defined within the deployment descriptor. This merge is performed by the Annotations Metadata Manager (AMM) facility. When the metadata is merged, the following inheritance rules are followed.

Table 132. Metadata merger inheritance rules.

This table lists the metadata merger inheritance rules.

Scenario	Rules
Security metadata in deployment descriptor only	No merge is needed, the security metadata from the deployment descriptor is propagated.
Security metadata in annotations only	No merge is needed, the security metadata defined with annotations is propagated.
Security metadata in deployment descriptor and annotations	The metadata from the deployment descriptor and annotations is merged. The metadata in annotations is overridden by the same type of data from the deployment descriptor.

Six security annotations are currently supported. For each annotation, a MergeAction implementation is defined.

- @DeclareRoles (Servlet 2.5 and greater and EJB 3)

  The MergeAction implementation finds all the classes annotated with the DeclareRoles annotation. Within each annotated class for each role name specified, if the security roles listed in the deployment descriptor does not contain a SecurityRole with the annotated role name, a new SecurityRole is created and added to this list of security roles.
- @RunAs (Servlet 2.5 and greater and EJB 3)

  The MergeAction implementation finds all the classes with the RunAs annotation. For each annotated class, it finds the Servlet or the Enterprise Java Bean (EJB) associated with the given class. It then determines if a run-as element is defined in the deployment descriptor for the servlet or EJB. If one is not found, a new run-as element is created and added to the deployment descriptor. If a run-as element is found, this run-as element will be used instead of creating a new one. The role name used in the RunAs annotation must be defined in the deployment descriptor.
- @DenyAll (EJB 3 only)

  The MergeAction implementation finds all the methods annotated with the DenyAll annotation. For each annotated method, if the method is not included in the deployment descriptor list of excluded methods, and a MethodPermission does not exist in the deployment descriptor, a new MethodElement is created and added to this list of excluded methods in the deployment descriptor.

- @PermitAll (EJB 3 only)

  The MergeAction implementation finds all the classes and the methods with the PermitAll annotation. For each annotated class, it finds the Enterprise Java Bean (EJB) associated with the given class. It then searches the subset of the MethodElements in the list of all the MethodPermissions defined in the deployment descriptor for this EJB. If a MethodElement with a wildcard method name ("*") is not found and a wildcard method does not exist in the list of excluded methods or in the list of MethodElements with security roles, a new MethodPermission and a new MethodElement are created. The new MethodPermission is marked unchecked and is added to the MethodPermission list in the deployment descriptor. The new MethodElement is added to the MethodElement list of the newly created unchecked MethodPermission. Similar action is done for all annotated methods. Instead of a wildcard MethodElement, the method signature must match exactly the signature of the annotated method.
- @RolesAllowed (EJB 3 only)

  The MergeAction implementation finds all of the classes and methods with the RolesAllowed annotation. For each annotated class, it finds the EJB associated with the given class. It then finds the subset of the MethodElements in the list of all the MethodPermissions defined in the deployment descriptor for this EJB. If a MethodElement with a wildcard method name ("*") is not found, and a wildcard method does not exist in the list of excluded methods or in the list of unchecked MethodElements, a new MethodPermission and MethodElement are created. If a MethodPermission for this EJB exists with exactly the same roles as those found in the annotation, this MethodPermission will be used instead of creating a new one. For each role name specified in the annotation, a new SecurityRole is created and added to the SecurityRole list in the MethodPermission, If the MethodPermission was newly created, it is added to the MethodPermission list in the deployment descriptor. The new MethodElement created is added to the MethodElement list of the MethodPermission. Similar processing is done for all annotated methods. Instead of a wildcard MethodElement, the method signature must exactly match the signature of the annotated method. Additionally, for each role name specified in the annotation, if the deployment descriptor list of security roles does not contain a SecurityRole with the annotated role name, a new SecurityRole is also created and added to this list of security roles.
- @ServletSecurity (Servlet 3.0 only)

  Note: Support for ServletSecurity annotation for Servlet 3.0 is new in this release of WebSphere Application Server.

  When an application deploys, the ServletSecurity MergeAction implementation finds all servlets with the ServletSecurity annotation. For each annotated servlet, it finds the servlet associated with the given class base on the WebServlet annotation. If RolesAllowed in the ServletSecurity annotation is not found in the deployment descriptor, it then creates a role-name attribute for the role in the deployment descriptor.

  When an application starts, the WebContainer inspects all servlets with the RunAs, declareRoles, and ServletSecurity annotations, and sets those annotations on the setServletSecurity() method of the ServletRegistration annotation. The WebContainer notifies the security component to inspect all ServletRegistration annotations that have URL patterns and security constraints. The security component then determines if a URL pattern is defined in the deployment descriptor. If one is not defined in the deployment descriptor, the security constraints and RunAs role in the URL pattern are created and then used. If an exact match is already defined in the deployment descriptor, the security constraints and RunAs role in the URL pattern of the deployment descriptor are used instead of the annotation data.

  Note: When the web authentication system property, com.ibm.wsspi.security.web.webAuthReq, is set to `persisting`, you can log into an unprotected URL if a valid username and password are provided.

  Note: ServletSecurity MergeAction and WebContainer require the WebServlet annotation to determine the servlet name of the URL pattern. The ServletSecurity annotation is omitted if the WebServlet annotation is missing, or if a URL pattern is not specified in the ServletSecurity annotation.

The Inherited servlet annotation is a metadata annotation. Do not specify the Inherited annotation in the class. If a subclass does not have security annotation, it automatically inherits security annotation from the parent class. The subclass can overwrite the parent security annotations by specifying its security annotations.

The following example is for all HTTP methods with no constraints:

```
@WebServlet ("/Example")
@ServletSecurity
public class Example extends HttpServlet {
       ......
}
```

The following example is for all HTTP methods with no <auth-constraint> element and confidential TransportGuarantee required:

```
@WebServlet ("/Example")
@ServletSecurity(@HttpConstraint(transportGuarantee =
                     TransportGuarantee.CONFIDENTIAL))
public class Example extends HttpServlet {
       ......
}
```

The following example is for all HTTP methods with all access denied:

```
@WebServlet ("/Example")
@ServletSecurity(@HttpConstraint(EmptyRoleSemantic.DENY))
public class Example extends HttpServlet {
       ......
}
```

The following example is for all HTTP methods except for the GET and POST values with no constraints. For GET, the <auth-constraint> element requires membership in ALL ROLE. For POST, all access is denied.

```
@WebServlet (name="Example", urlPatterns={"/Example"})
@ServletSecurity((httpMethodConstraints = {
 @HttpMethodConstraint(value = "GET", rolesAllowed = "ALL ROLE"),
 @HttpMethodConstraint(value="POST", emptyRoleSemantic =
EmptyRoleSemantic.DENY))
})
public class Example extends HttpServlet {
       ......
}
```

The following example is for all HTTP methods except GET, the <auth-constraint> element requires membership in ALL ROLE, and the GET method has no constraints.

```
@WebServlet (name="Example", urlPatterns={"/Example"})
@ServletSecurity(value = @HttpConstraint(rolesAllowed = "ALL ROLE"),
 httpMethodConstraints = @HttpMethodConstraint("GET"))
public class Example extends HttpServlet {
       ......
}
```

The following example is for all HTTP methods except TRACE, the <auth-constraint> element requires membership in ALL ROLE, and for TRACE, all access is denied.

```
@WebServlet (name="Example", urlPatterns={"/Example"})
@ServletSecurity(value = @HttpConstraint(rolesAllowed = "ALL ROLE"),
 httpMethodConstraints = @HttpMethodConstraint(value="TRACE",
 emptyRoleSemantic = EmptyRoleSemantic.DENY))
public class Example extends HttpServlet {
       ......
}
```

Java Servlet 3.0 support for security:

This release of WebSphere Application Server supports all security updates as defined in the Java Servlet 3.0 specification.

Note: This release of WebSphere Application Server supports all security updates as defined in the Java Servlet 3.0 specification (JSR-315), including the new servlet security annotations, use of new programmatic security APIs and the dynamic updating of the servlet security configuration.

A significant enhancement is the new annotation support for servlets. A developer can declare the security constraints using annotations as an alternative to declaring them as part of the web.xml file, which is used prior to Java Servlet 3.0. The `web.xml` file continues to function and overrides any conflicts defined as annotations.

The list of supported Java Servlet 3.0 updates for security includes the following:

- Support for the @ServletSecurity annotation
- Support for the dynamic updating of the @RunAs, @declareRoles, and @ServletSecurity servlet security annotations
- Support for the authenticate, login and logout servlet security methods
- The new com.ibm.websphere.security.displayRealm property specifies whether the HTTP basic authentication login window displays the realm name that is not defined in the application `web.xml` file.

The following discusses the Java Servlet 3.0 updates for security in more detail:

Support for the @ServletSecurity annotation:

When an application deploys, the ServletSecurity MergeAction implementation finds all servlets with the ServletSecurity annotation. For each annotated servlet, it finds the servlet associated with the given class base on the WebServlet annotation. If RolesAllowed in the ServletSecurity annotation is not found in the deployment descriptor, it then creates a role-name attribute for the role in the deployment descriptor.

When an application starts, the WebContainer inspects all servlets with the RunAs, declareRoles, and ServletSecurity annotations, and sets those annotations on the setServletSecurity() method of the ServletRegistration annotation. The WebContainer notifies the security component to inspect all ServletRegistration annotations that have URL patterns and security constraints. The security component then determines if a URL pattern is defined in the deployment descriptor. If one is not defined in the deployment descriptor, the security constraints and RunAs role in the URL pattern are created and then used. If an exact match is already defined in the deployment descriptor, the security constraints and RunAs role in the URL pattern of the deployment descriptor are used instead of the annotation data.

Read the Security annotations topic for more information.

Support for the dynamic updating of the @RunAs, @declareRoles, and @ServletSecurity servlet security annotations:

When an application starts, the web container inspects all servlets with the RunAs, declareRoles, and ServletSecurity annotations, and sets those annotations on the setServletSecurity() method of the ServletRegistration annotation. The web container notifies the security component to inspect all ServletRegistration annotations that have URL patterns and security constraints. The security component then determines if a URL pattern is defined in the deployment descriptor. If an exact match is already defined in the deployment descriptor, the security constraints and RunAs role in the URL pattern of the deployment descriptor are used instead of the dynamic data.

Read the Servlet security dynamic annotations topic for more information.

Note: WebSphere Application Server supports both a default authorization provider and an authorization provider that is based on the Java Authorization Contract for Containers (JACC) specification. The JACC-based authorization provider (for example, the Tivoli Access Manager), enables third-party security providers to handle the Java EE authorization. The RunAs, declareRoles, and ServletSecurity annotations are supported for both native authorization and for JACC.

Support for the authenticate, login and logout servlet security methods:

The authenticate method authenticates a user by using the WebSphere Application Server container login mechanism configured for the servlet context.

The login method authenticates a user to the WebSphere Application Server with a user ID and password. If authentication is successful, it creates a user subject on the thread and Lightweight Third Party Authentication (LTPA) cookies (if single sign-on (SSO) is enabled).

The logout method logs the user out of the WebSphere Application Server and invalidates the HTTP session.

Read the Servlet security methods topic for more information.

The new com.ibm.websphere.security.displayRealm property specifies whether the HTTP basic authentication login window displays the realm name that is defined in the application `web.xml` file:

If the realm name is not defined in the `web.xml` file, one of the following occurs:

- If the property is set to `false` (the default), the WebSphere realm name display is Default Realm.
- If the property is set to `true`, the WebSphere realm name display is the user registry realm name for the LTPA authentication mechanism or the Kerberos realm name for the Kerberos authentication mechanism.

Read the Security custom properties topic for more information.

Servlet security dynamic annotations:

When you use the programmatic APIs to add or to create a servlet, the security annotations, RunAs, declareRoles and ServletSecurity, can be dynamically updated through the setRunAsRole(), declareRoles() and setServletSecurity() methods respectively.

Note: Support for the dynamic updating of the RunAs, declareRoles, and ServletSecurity servlet security annotations is new in this release of WebSphere Application Server.

When an application starts, the web container inspects all servlets with the RunAs, declareRoles, and ServletSecurity annotations, and sets those annotations on the setServletSecurity() method of the ServletRegistration annotation. The web container notifies the security component to inspect all ServletRegistration annotations that have URL patterns and security constraints. The security component then determines if a URL pattern is defined in the deployment descriptor. If an exact match is already defined in the deployment descriptor, the security constraints and RunAs role in the URL pattern of the deployment descriptor are used instead of the dynamic data.

Note: If the dynamic security annotations, declareRoles, setRunAs and rolesAllowed, are used, the role name must be pre-defined, either through the deployment descriptor or through the declareRoles and or RunAs annotations in the servlet class. During deployment time, you can use the administrative console to map a user or group to this role.

If you have an exact URL pattern match for the ServletSecurity annotation in the security dynamic annotation, the security constraint of the URL pattern in the security dynamic annotation takes precedent. Also, if you call the setServletSecurity() method multiple times with the same URL pattern, the last one takes precedent.

- ServletRegistration.Dynamic.setRunAsRole(String roleName) sets the name of the RunAs role for this servlet registration.
- ServletContext.declareRoles(String roleNames) declares role names that are tested for the isUserInRole() method.
- ServletRegistration.Dynmaic.setServletSecurity(ServletSecurityElement constraint) sets the ServletSecurityElement for this servlet registration.

Note: When the web authentication system property, com.ibm.wsspi.security.web.webAuthReq, is set to `persisting`, you can log into an unprotected URL if a valid username and password are provided.

The following two examples can be used to set the security constraints and RunAs role for dynamic servlets by using the setServletSecurity() method.

In this example, all HTTP elements require membership in the Employee role except for the PUT method. For the PUT method, the <auth-constraint> element requires membership in the Manager role and TransportGuarantee is confidential.

```
HttpConstraintElement constraint = new HttpConstraintElement(TransportGuarantee.NONE,
new String[]{"Employee"});
List<HttpMethodConstraintElement> methodConstraints =
new ArrayList<HttpMethodConstraintElement>();
methodConstraints.add(new HttpMethodConstraintElement("PUT",
new HttpConstraintElement(TransportGuarantee.CONFIDENTIAL, new String[]{"Manager"})));
ServletSecurityElement servletSecurity =
new ServletSecurityElement(constraint, methodConstraints);
```

In this example, all HTTP methods are allowed except for the CUSTOM and GET methods. For the CUSTOM method, the <auth-constraint> element requires membership in the Manager role. For the GET method, the <auth-constraint> element requires membership in the Employee role, and TransportGuarantee is confidential.

```
HttpConstraintElement constraint = new HttpConstraintElement();
List<HttpMethodConstraintElement> methodConstraints =
new ArrayList<HttpMethodConstraintElement>();
methodConstraints.add(new HttpMethodConstraintElement("CUSTOM",
new HttpConstraintElement(TransportGuarantee.NONE, new String[]{"Manager"})));
methodConstraints.add(new HttpMethodConstraintElement("GET",
new HttpConstraintElement(TransportGuarantee.CONFIDENTIAL, new String[]{"Employee"})));
ServletSecurityElement servletSecurity = new ServletSecurityElement(constraint,
methodConstraints);
```

Delegations

Delegation is a process security identity propagation from a caller to a called object. As per the Java Platform, Enterprise Edition (Java EE) specification, a servlet and enterprise beans can propagate either the client or remote user identity when invoking enterprise beans, or they can use another specified identity as indicated in the corresponding deployment descriptor.

The extension supports enterprise bean propagation to the server ID when invoking other entity beans. Three types of delegations are possible:

- Delegate (RunAs) client identity
- Delegate (RunAs) specified identity
- Delegate (RunAs) system identity

Note: The RunAs system identity delegation only works when server ID and password are used. When the internalServerId feature is used, it does not work because runAs with system identity is not supported. You must specify RunAs roles. When internalServerID is used, use the RunAsSpecified with a user ID and password that is mapped to the administrator role. See “Administrative roles and naming service authorization” on page 1615 for more information about internalServerId.

The EJB specification only supports delegation (RunAs) at the Enterprise JavaBeans (EJB) level. But an extension allows EJB method-level RunAs specification. With an EJB method level, the RunAs specification can specify a different RunAs role for different methods within the same enterprise beans.

The RunAs specification is detailed in the deployment descriptor, which is the `ejb-jar.xml` file in the EJB module and the `web.xml` file in the web module. The extension to the RunAs specification is included in the `ibm-ejb-jar-ext.xml` file.

An IBM-specific binding file is available for each application that contains a mapping from the RunAs role to the user. This file is specified in the `ibm-application-bnd.xml` file.

These specifications are read by the runtime during application startup. The following figure illustrates the delegation mechanism, as implemented in the WebSphere Application Server security model.

Delegation Process

Two tables help in the delegation process:
- Resource to RunAs role mapping table
- RunAs role to user ID and password mapping table

Use the Resource to RunAs role mapping table to get the role that is used by a servlet or by enterprise beans to propagate to the next enterprise beans call.

Use the RunAsRole to user ID and password mapping table to get the user ID that belongs to the RunAs role and its password.

Delegation is performed after successful authentication and authorization. During this process, the delegation module consults the Resource to RunAs role mapping table to get the RunAs role (3). The delegation module consults the RunAs role to user ID and password mapping table to get the user that belongs to the RunAs role (4). The user ID and password is used to create a new credential using the authentication module, which is not shown in the figure.

The resulting credential is stored in the Object Request Broker (ORB) Current as an invocation credential (5). Servlet and enterprise beans when invoking other enterprise beans pick up the invocation credential from the ORB Current (6) and call the next enterprise beans (7).

Authorizing access to Java EE resources using Tivoli Access Manager

The Java Authorization Contract for Containers (JACC) defines a contract between Java Platform, Enterprise Edition (Java EE) containers and authorization providers. You can use the default authorization or an external JACC authorization provider. When security is enabled in WebSphere Application Server, the default authorization is used unless a JACC provider is specified.

Before you begin

JACC enables any third-party authorization providers to plug into a Java EE application server (such as WebSphere Application Server) to make the authorization decisions when a Java EE resource is accessed. By default, WebSphere Application Server implements the JACC provider by using Tivoli Access Manager as the external authorization provider.

Read the following articles for more detailed information about JACC before you attempt to configure WebSphere Application Server to use a JACC provider:

Procedure

- “JACC support in WebSphere Application Server” on page 1627
- “JACC providers” on page 1630
- “Tivoli Access Manager integration as the JACC provider” on page 1634

Using the built-in authorization provider

You can extend the capabilities of WebSphere Application Server by plugging in your own authorization provider. You can use the built-in authorization or an external JACC authorization provider.

About this task

For an explanation of the administrative console panels that support these capabilities, see:

Procedure

- Use the built-in authorization provider. It is recommended that you do not modify any settings on the authorization provider panels if you use the **Built-in authorization** option. For more information, see "External authorization provider settings."
- Use an external authorization provider. If you use the **External authorization using a JACC provider** option, the external providers must be based on the Java Authorization Contract for Containers (JACC) specification to handle the Java Platform, Enterprise Edition (Java EE) authorization. By default, WebSphere Application Server enables you to configure the Tivoli Access Manager Java Authorization Contract for Containers (JACC) provider as the default external JACC provider. For more information, see "External Java Authorization Contract for Containers provider settings" on page 1646 and "Tivoli Access Manager JACC provider settings" on page 1652.

External authorization provider settings:

Use this page to enable a Java Authorization Contract for Containers (JACC) provider for authorization decisions.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Click **External authorization providers**.

The application server provides a default authorization engine that performs all of the authorization decisions. In addition, the application server also supports an external authorization provider using the JACC specification to replace the default authorization engine for Java Platform, Enterprise Edition (Java EE) applications.

JACC is part of the Java EE specification, which enables third-party security providers such as Tivoli Access Manager to plug into the application server and make authorization decisions.

Important: Unless you have an external JACC provider or want to use a JACC provider for Tivoli Access Manager that can handle Java EE authorizations based on JACC, and it is configured and set up to use with the application server, do not enable **External authorization using a JACC provider**.

Built-in authorization:

Use this option all the time unless you want an external security provider such as the Tivoli Access Manager to perform the authorization decision for Java EE applications that are based on the JACC specification.

External JACC provider: Use this link to configure the application server to use an external JACC provider. For example, to configure an external JACC provider, the policy class name and the policy configuration factory class name are required by the JACC specification.

The default settings that are contained in this link are used by Tivoli Access Manager for authorization decisions. If you intend to use another provider, modify the settings as appropriate.

External Java Authorization Contract for Containers provider settings:

Use this page to configure the application server to use an external Java Authorization Contract for Containers (JACC) provider. For example, the policy class name and the policy configuration factory class name are required by the JACC specification.

Use these settings when you have set up an external security provider that supports the JACC specification to work with the application server. The configuration process involves installing and configuring the provider server and configuring the client of the provider in the application server to communicate with the server. If the JACC provider is not enabled, these settings will be ignored.

To view this administrative console page, complete the following steps:

1. Click **Security > Global security**.
2. Click **External authorization providers**.
3. Under Authorization provider, click **External JACC provider**.

Use the default settings when you use Tivoli Access Manager as the JACC provider. Install and configure the Tivoli Access Manager server prior to using it with the application server. Use the Tivoli Access Manager properties link under Additional properties, and configure the Tivoli Access Manager client in the application server to use the Tivoli Access Manager server. If you intend to use another provider, modify the settings as appropriate.

Name:

Specifies the name that is used to identify the external JACC provider.

This field is required.

Data type: String

Description:

Provides an optional description for the provider.

Data type: String

Policy class name:

Specifies a fully qualified class name that represents the javax.security.jacc.policy.provider property as per the JACC specification. The class represents the provider-specific implementation of the java.security.Policy abstract methods.

The class file must reside in the class path of each application server process. These processes include the application server, node agents and the deployment manager.

Do not add the Java archive (JAR) file, which contains the class file, to the `<WAS_HOME>/lib` directory in a product environment as service releases overwrite files in this directory.

This class is used during authorization decisions. The default class name is for Tivoli Access Manager implementation of the policy file.

This field is required. For information on enabling the JACC provider using this field, see the "Enabling the JACC provider for Tivoli Access Manager" article in the information center.

Data type: String
Default: com.tivoli.pd.as.jacc.TAMPolicy

Policy configuration factory class name:

Specifies a fully qualified class name that represents the javax.security.jacc.PolicyConfigurationFactory.provider property as per the JACC specification. The class represents the provider-specific implementation of the javax.security.jacc.PolicyConfigurationFactory abstract methods.

The class file must reside in the class path of each application server process. These processes include the application server, node agents and the deployment manager.

Do not add the Java archive (JAR) file, which contains the class file, to the `<WAS_HOME>/lib` directory in a product environment as service releases overwrite files in this directory.

This class represents the provider-specific implementation of the PolicyConfigurationFactory abstract class. This class is used to propagate the security policy information to the JACC provider during the installation of the J2EE application. The default class name is for the Tivoli Access Manager implementation of the policy configuration factory class name.

This field is required.

Data type: String
Default: com.tivoli.pd.as.jacc.TAMPolicyConfigurationFactory

Role configuration factory class name:

Specifies a fully qualified class name that implements the com.ibm.wsspi.security.authorization.RoleConfigurationFactory interface.

The class file must reside in the class path of each application server process. These processes include the application server, node agents and the deployment manager.

Do not add the Java archive (JAR) file, which contains the class file, to the `<WAS_HOME>/lib` directory in a product environment as service releases overwrite files in this directory.

When you implement this class, the authorization table information in the binding file is propagated to the provider during the installation of the J2EE application. The default class name is for the Tivoli Access Manager implementation of the role configuration factory class name.

This field is optional. For information on enabling the JACC provider using this field, see the "Enabling the JACC provider for Tivoli Access Manager" article in the information center.

Data type: String
Default: com.tivoli.pd.as.jacc.TAMRoleConfigurationFactory

Provider initialization class name:

Specifies a fully qualified class name that implements the com.ibm.wsspi.security.authorization.InitializeJACCProvider interface.

The class file must reside in the class path of each application server process. These processes include the application server, node agents and the deployment manager.

Do not add the Java archive (JAR) file, which contains the class file, to the <WAS_HOME>/lib directory in a product environment as service releases overwrite files in this directory.

When implemented, this class is called at the start and the stop of all the application server processes. You can use this class for any required initialization that is needed by the provider client code to communicate with the provider server. The properties that are entered in the custom properties link are passed to the provider when the process starts up. The default class name is for the Tivoli Access Manager implementation of the provider initialization class name.

This field is optional. For information on enabling the JACC provider using this field, see the "Enabling the JACC provider for Tivoli Access Manager" article in the information center.

Data type: String
Default: com.tivoli.pd.as.jacc.cfg.TAMConfigInitialize

Requires the EJB arguments policy context handler for access decisions:

Specifies whether the JACC provider requires the EJBArgumentsPolicyContextHandler handler to make access decisions.

Because this option has an impact on performance, do not set it unless it is required by the provider. Normally, this handler is required only when the provider supports instance-based authorization. Tivoli Access Manager does not support this option for J2EE applications.

Default: Disabled

Supports dynamic module updates:

Specifies whether you can apply changes made to security policies of web modules in a running application, dynamically without affecting the rest of the application.

If this option is enabled, the security policies of the added or modified web modules are propagated to the JACC provider and only the affected web modules are started.

If this option is disabled, then the security policies of the entire application are propagated to the JACC provider for any module-level changes. The entire application is restarted for the changes to take effect.

Typically, this option is enabled for an external JACC provider.

Default: Enabled

Custom properties:

Specifies the properties that are required by the provider.

These properties are propagated to the provider during the startup process when the provider initialization class name is initialized. If the provider does not implement the provider initialization class name as described previously, the properties are not used.

The Tivoli Access Manager implementation does not require that you enter any properties in this link.

Tivoli Access Manager properties:

Specifies properties that are required by the Tivoli Access Manager implementation.

These properties are used to set up the communication between the application server and the Tivoli Access Manager server. You must install and configure the Tivoli Access Manager server before entering these properties.

Enabling an external JACC provider

Use this topic to enable an external JACC provider using the administrative console.

Before you begin

The Java Authorization Contract for Containers (JACC) defines a contract between Java Platform, Enterprise Edition (Java EE) containers and authorization providers. This contract enables any third-party authorization providers to plug into a Java EE 5 application server, such as WebSphere Application Server to make the authorization decisions when a Java EE resource is accessed.

Procedure

1. From the WebSphere Application Server administrative console, click **Security > Global security > External authorization providers**.
2. Under Related items, click **External JACC provider**.
3. The fields are set for Tivoli Access Manager by default. If you do not plan to use Tivoli Access Manager as the JACC provider, replace these fields with the details for your own external JACC provider.
4. If any custom properties are required by the JACC provider, click **Custom properties** under Additional properties and enter the properties. When using the Tivoli Access Manager, use the **Tivoli Access Manager properties** link instead of the Custom properties link. For more information, see "Configuring the JACC provider for Tivoli Access Manager using the administrative console."
5. On the External authorization providers panel, select the **External authorization using a JACC provider** option and click **OK**.
6. Complete the remaining steps to enable security. If you are using Tivoli Access Manager, you must select LDAP as the user registry and use the same LDAP server. For more information on configuring LDAP registries, see "Configuring Lightweight Directory Access Protocol user registries" on page 1260.
7. Restart all servers to make these changes effective.

Configuring the JACC provider for Tivoli Access Manager using the administrative console:

Use this task to configure Tivoli Access Manager as the Java Authorization Contract for Containers (JACC) provider using the administrative console.

Before you begin

Prior to completing the following steps, verify that you have previously created a security administrative user. For more information, see "Creating the security administrative user for Tivoli Access Manager" on page 1651.

About this task

The following configuration is performed on the management server. When you click either **Apply** or **OK**, configuration information is checked for consistency, saved, and applied if successful.

To configure Tivoli Access Manager as the JACC provider using the administrative console, complete the following steps:

Procedure

1. Start the WebSphere Application Server administrative console by clicking `http://`*`yourhost`*`.domain:`*`port_number`*`/ibm/console` after starting WebSphere Application Server. If security is

currently disabled, log in with any user ID. If security is currently enabled, log in with a predefined administrative ID and password. This ID is typically the server user ID that is specified when you configure the user registry.

2. Click **Security > Global security > External authorization providers**.
3. Under General properties, select **External authorization using a JACC provider**.
4. Under Related items, click **External JACC provider**.
5. Under Additional properties, click **Tivoli Access Manager Properties**. The Tivoli Access Manager JACC provider configuration screen is displayed.
6. Enter the following information:

   Enable embedded Tivoli Access Manager
   Select this option to enable Tivoli Access Manager.

   Ignore errors during embedded Tivoli Access Manager disablement
   Select this option when you want to unconfigure the JACC provider. Do not select this option during configuration.

   Client listening port set
   WebSphere Application Server must listen using a TCP/IP port for authorization database updates from the policy server. More than one process can run on a particular node or machine. More than one authorization server can be specified by separating the entries with commas. Specifying more than one authorization server at a time is useful for reasons of failover and performance. Enter the listening ports used by Tivoli Access Manager clients, separated by a comma. If a range of ports is specified, separate the lower and higher values by a colon (**:**) (for example, `7999, 9990:999`).

   Policy server
   Enter the name of the Tivoli Access Manager policy server and the connection port. Use the `policy_server:port` form. The policy communication port is set at the time of the Tivoli Access Manager configuration, and the default is 7135.

   Authorization servers
   Enter the name of the Tivoli Access Manager authorization server. Use the `auth_server:port:priority` form. The authorization server communication port is set at the time of the Tivoli Access Manager configuration, and the default is 7136. The priority value is determined by the order of the authorization server use (for example, `auth_server1:7136:1` and `auth_server2:7137:2`). A priority value of `1` is required when configuring against a single authorization server.

   Administrator user name
   Enter the Tivoli Access Manager administrator user name that was created when Tivoli Access Manager was configured; it is usually `sec_master`.

   Administrator user password
   Enter the Tivoli Access Manager administrator password.

   User registry distinguished name suffix
   Enter the distinguished name suffix for the user registry that is shared between Tivoli Access Manager and WebSphere Application Server, for example, `o=ibm, c=us`.

   Security domain
   You can create more than one security domain in Tivoli Access Manager, each with its own administrative user. Users, groups and other objects are created within a specific domain, and are not permitted to access resource in another domain. Enter the name of the Tivoli Access Manager security domain that is used to store WebSphere Application Server users and groups.

   If a security domain is not established at the time of the Tivoli Access Manager configuration, leave the value as `Default`.

Administrator user distinguished name

Enter the full distinguished name of the WebSphere Application Server security administrator ID (for example, `cn=wasdmin, o=organization, c=country`). The ID name must match the Server user ID on the Lightweight Directory Access Protocol (LDAP) User Registry panel in the administrative console. To access the LDAP User Registry panel, click **Security > Global security**. Under **User account repository**, choose **Standalone LDAP registry** as the available realm definition. Then click **Configure**.

7. When all information is entered, click **OK** to save the configuration properties. The configuration parameters are checked for validity and the configuration is attempted at the host server or cell manager.

Results

After you click **OK**, WebSphere Application Server completes the following actions:

- Validates the configuration parameters.
- Configures the host server or cell manager.

These processes might take some time depending on network traffic or the speed of your machine.

What to do next

If the configuration is successful, the parameters are copied to all subordinate servers, including the node agents. To complete the embedded Tivoli Access Manager client configuration, you must restart all of the servers, including the host server, and enable WebSphere Application Server security.

Creating the security administrative user for Tivoli Access Manager:

Enabling security requires the creation of a WebSphere Application Server administrative user. Use the Tivoli Access Manager command-line pdadmin utility to create the Tivoli Access Manager administrative user for WebSphere Application Server. This utility is available on the policy server host machine.

About this task

Follow these steps to use the pdadmin utility.

Procedure

1. From a command line, start the pdadmin utility as the Tivoli Access Manager administrative user, sec_master:

```
pdadmin -a sec_master -p sec_master_password
```

2. Create a WebSphere Application Server security user. For example, the following instructions create a new user, *wasadmin*. The command is entered as one continuous line:

```
pdadmin> user create wasadmin cn=wasadmin,o=organization,
c=country wasadmin wasadmin myPassword
```

   Substitute values for organization and country that are valid for your Lightweight Directory Access Protocol (LDAP) user registry.

3. Enable the account for the WebSphere Application Server security administrative user by issuing the following command:

```
pdadmin> user modify wasadmin account-valid yes
```

What to do next

Configure the Java Authorization Contract for Container (JACC) provider for Tivoli Access Manager. For more information, see “Tivoli Access Manager JACC provider configuration.”

Tivoli Access Manager JACC provider configuration:

You can configure the Java Authorization Contract for Containers (JACC) provider for Tivoli Access Manager to deliver authentication and authorization protection for your applications or for authentication only. Most deployments that use the JACC provider for Tivoli Access Manager to configure Tivoli Access Manager provide both authentication and authorization functionality.

If you want Tivoli Access Manager to provide authentication, but leave authorization as part of WebSphere Application Server's native security, add the `com.tivoli.pd.as.amwas.DisableAddAuthorizationTableEntry=true` property to the `amwas.amjacc.template.properties` file. The file is located in the *profile_root*`/config/cells/`*cell_name* directory.

After this property is set, perform the tasks for setting Tivoli Access Manager Security, as documented.

You can configure the JACC provider for Tivoli Access Manager using either the WebSphere Application Server administrative console or the **wsadmin** command-line utility.

- For details on configuring the JACC provider for Tivoli Access Manager using the administrative console, refer to "Configuring the JACC provider for Tivoli Access Manager using the administrative console" on page 1649.
- For details on configuring the Tivoli Access Manager JACC provider using the **wsadmin** command line utility, refer to Configuring the JACC provider for Tivoli Access Manager using the wsadmin utility.

The JACC configuration files are not common across multiple WebSphere Application profiles. The following property setting is added to the *profile_root*`/config/cells/`*cell_name*`/amwas.amjacc.template.properties` file to specify the location of the JACC configuration for each profile.

```
com.tivoli.pd.as.jacc.CommonFileLocation=USER_INSTALL_ROOT/etc/pd
```

The **wsadmin** command is available to reconfigure the Java Authorization Contract for Containers (JACC) Tivoli Access Manager interface:

```
$AdminTask reconfigureTAM -interactive
```

This command effectively prompts you through the process of unconfiguring the interface and then reconfiguring it.

Tivoli Access Manager JACC provider settings:

Use this page to configure the Java Authorization Contract for Container (JACC) provider for Tivoli Access Manager.

Note: When a third-party authorization such as Tivoli Access Manager or SAF for z/OS is used, the information in the administrative console panel might not represent the data in the provider. Also, any changes to the panel might not be reflected in the provider automatically. Follow the provider's instructions to propagate any changes made to the provider.

To view the JACC provider settings for Tivoli Access Manager, complete the following steps:

1. Click **Security > Global security**.
2. Under Authentication, click **External authorization providers**.
3. Under Authorization provider, click **External JACC provider**.
4. Click **Configure** to configure the properties for Tivoli Access Manager.

Enable embedded Tivoli Access Manager:

Enables or disables the embedded Tivoli Access Manager client configuration.

Default: Disabled

Range: Enabled or Disabled

Note: If you want to disable Tivoli Access Manager as the JACC provider, clear this option and also select **Default authorization**.

Ignore errors during embedded Tivoli Access Manager disablement:

Specifies whether to ignore error messages during the unconfiguration process.

If you check this check box and click **OK** or **Apply**, when you unconfigure the embedded Tivoli Access Manager, any unconfiguration errors are ignored and the process completes. If you do not check this check box, unconfiguration errors cause the unconfiguration process to stop.

This option is applicable only when re-configuring an embedded Tivoli Access Manager client or disabling an embedded Tivoli Access Manager.

Default: Disabled
Range: Enabled or Disabled

Client listening port set:

Enter the ports that are used as listening ports by Tivoli Access Manager clients.

The application server needs to listen on a TCP/IP port for authorization database updates from the policy server. More than one process can run on a particular node and machine, so a list of ports is required for use by the processes. If you specify a range of ports, separate the lower and higher values by a colon (:). The first 20% of the range is reserved for the deployment manager. Single ports and port ranges are specified on separate lines. An example list might look like the following example:

```
7999
 8900:8999
```

Policy server:

Enter the name, fully-qualified domain name, or IP address of the Tivoli Access Manager policy server and the connection port.

Use the form *policy_server*:*port*. The policy server communication port was set at the time of the Tivoli Access Manager configuration. The default is 7135.

Authorization servers:

Enter the name, fully-qualified domain name, or IP address of the Tivoli Access Manager authorization server. Use the form, *auth_server*:*port*:*priority*.

The authorization server communication port is set at the time of Tivoli Access Manager configuration. The default is 7136. You can specify more than one authorization server by entering each server on a new line. Configuring more than one authorization server provides for failover. The priority value is the order of authorization server use. For example:

```
auth_server1.mycompany.com:7136:1
auth_server2.mycompany.com:7137:2
```

A priority of 1 is still required when configuring a single authorization server.

Administrator user name:

Enter the Tivoli Access Manager administration user ID, as created at the time of Tivoli Access Manager configuration. This ID is usually, `sec_master`.

Administrator user password:

Enter the Tivoli Access Manager administration password for the user ID that is entered in the **Administrator user name** field.

User registry distinguished name suffix:

Enter the distinguished name suffix for the user registry to share between Tivoli Access Manager and the application server. For example: o=*organization*,c=*country*

Security domain:

Enter the name of the Tivoli Access Manager security domain that is used to store application server users and groups.

Specification of the Tivoli Access Manager domain is required because more than one security domain can be created in Tivoli Access Manager with its own administrative user. Users, groups, and other objects are created within a specific domain and are not permitted to access resources in another domain. If a security domain is not established at the time of Tivoli Access Manager configuration, leave the value as *Default*.

Default: Default

Administrator user distinguished name:

Enter the fully distinguished name of the security administrator ID for the application server. For example, cn=wasadmin,o=*organization*,c=*country*

JACC provider configuration properties for Tivoli Access Manager:

The JACC provider configuration properties detailed below may require configuration.

The Java property files are created in the WebSphere Application Server *profile_root* directory.

Two properties files might require configuration:

- `amwas.`*`node_name_server_name`*`.amjacc.properties` contains properties that are used by the JACC provider of Tivoli Access Manager.
- `amwas.`*`node_name_server_name`*`.pdjlog.properties` contains logging properties that are created from the `amwas.pdjlog.template.properties` file for the specific node and server combination at the time of configuration.

Use `amwas.`*`node_name_server_name`*`.amjacc.properties` file to configure static role caching, dynamic role caching, object caching, and role-based policy framework properties.

Static role caching properties:

The static role cache holds role memberships that do not expire.

These properties are in the *profile_root*`/etc/tam/amwas.`*`node_name_server_name`*`.amjacc.properties` file.

The *profile_root* directory is the value of the profilePath parameter at profile creation time.

com.tivoli.pd.as.cache.EnableStaticRoleCaching=true:

Enables or disables static role caching. Static role caching is enabled by default.

com.tivoli.pd.as.cache.StaticRoleCache=com.tivoli.pd.as.cache.StaticRoleCacheImpl:

This property holds the implementation class of the static role cache. You do not need to change this property, although the opportunity exists to implement your own cache, if necessary.

com.tivoli.pd.as.cache.StaticRoleCache.Roles=Administrator,Operator,Monitor,Deployer:

Defines the administration roles for WebSphere Application Server.

Tip: Enhance Application performance by adding the static roles: **CosNamingRead**, **CosNamingWrite**, **CosNamingCreate**, **CosNamingDelete**. These roles support for improved lookup performance within the application naming service.

Dynamic role caching properties:

The dynamic role cache holds role memberships that expire.

These properties are in the *profile_root*/etc/tam/amwas.*node_name_server_name*.amjacc.properties file.

The *profile_root* directory is the value of the profilePath parameter at profile creation time.

com.tivoli.pd.as.cache.EnableDynamicRoleCaching=true:

Enables or disables dynamic role caching. Dynamic role caching is enabled by default.

com.tivoli.pd.as.cache.DynamicRoleCache=com.tivoli.pd.as.cache.DynamicRoleCacheImpl:

This property holds the implementation class of the dynamic role cache. You do not need to change this property, although the opportunity exists to implement your own cache, if necessary.

com.tivoli.pd.as.cache.DynamicRoleCache.MaxUsers=100000:

The maximum number of users that the cache supports before a cache cleanup is performed. The default number of users is 100000.

com.tivoli.pd.as.cache.DynamicRoleCache.NumBuckets=20:

The number of tables that is used internally by the dynamic role cache. The default is 20. When a large number of threads use the cache, increase the value to tune and optimize cache performance.

com.tivoli.pd.as.cache.DynamicRoleCache.PrincipalLifeTime=10:

The period of time in minutes that a principal entry is stored in the cache. The default time is 10 minutes. The term, *principal*, here refers to the Tivoli Access Manager credential that is returned from a unique Lightweight Directory Access Protocol user.

com.tivoli.pd.as.cache.DynamicRoleCache.RoleLifetime=20:

The period of time in seconds that a role is stored in the role list for a user before it is discarded. The default is 20 seconds.

Object caching properties:

The object cache is used to cache all Tivoli Access Manager objects, including their extended attributes. This bypasses the need to query the Tivoli Access Manager authorization server for each resource request.

These properties are in the *profile_root*/etc/tam/amwas.*node_name_server_name*.amjacc.properties file.

The *profile_root* directory is the value of the profilePath parameter when the profile is created.

These object cache properties cannot be changed after configuration. If any require changing, it should be done before configuration of the nodes in the cell. Changes need to be made in the template properties file before any configuration actions are performed. Properties changed after configuration might cause access decisions to fail.

com.tivoli.pd.as.cache.EnableObjectCaching=true:

This property enables or disables object caching. The default value is true.

com.tivoli.pd.as.cache.ObjectCache=com.tivoli.pd.as.cache.ObjectCacheImpl:

This property is the class used to perform object caching. You can implement your own object cache if required. This can be done by implementing the *com.tivoli.pd.as.cache.IObjectCache* interface. The default is *com.tivoli.pd.as.cache.ObjectCacheImpl*.

com.tivoli.pd.as.cache.ObjectCache.NumBuckets=20:

This property specifies the number of buckets used to store object cache entries in the underlying hash table. The default is 20.

com.tivoli.pd.as.cache.ObjectCache.MaxResources=10000:

This property specifies the total number of entries for all buckets in the cache. This figure, divided by NumBuckets determines the maximum size of each bucket. The default is 10000.

com.tivoli.pd.as.cache.ObjectCache.ResourceLifeTime=20:

This property specifies the length of time in minutes that objects are kept in the object cache. The default is 20.

Role-based policy framework properties:

Although it is very unlikely that you will need to change these properties, use this file to reference supported properties within the role-based policy framework.

The role-based policy framework parameters are located in the Java Authorization Contract for Containers (JACC) configuration file and in the authorization configuration file. They are set at the time of JACC provider configuration and authorization server configuration. The role-based policy framework settings for the authorization table and the JACC provider can be modified separately for each WebSphere Application Server instance. The amwas.*node_server*.authztable.properties configuration file is generated from the authorization table. The amwas.*node_name_server_name*.amjacc.properties configuration file is generated from the JACC provider. Both files are stored in the *profile_root*/etc/tam directory. It is very unlikely that you might need to change these properties. The properties are described here for reference.

The settings cannot be changed after configuration. Make changes in the template properties file before any configuration actions are performed. Properties that are changed after configuration will cause access decisions to fail.

com.tivoli.pd.as.rbpf.AMAction=i:

This property is used to signify that a user is granted access to a role. This value is added to a Tivoli Access Manager access control list (ACL) and places invoke access on roles for users and groups.

com.tivoli.pd.as.rbpf.AMActionGroup=WebAppServer:

This property sets the Tivoli Access Manager action group that serves as a container for the action that is specified by the com.tivoli.pd.as.rbpf.AMAction property. The permission set in the `com.tivoli.pd.as.rbpf.AMAction` property goes into this action group.

com.tivoli.pd.as.rbpf.PosRoot=WebAppServer:

This property is used to determine where roles are stored in the protected object space.

com.tivoli.pd.as.rbpf.ProductId=deployedResources:

This property specifies the location under the root location that is specified in the posroot property to separate other products in the protected object space. Embedded Tivoli Access Manager objects are found in the `/WebAppServer/deployedResources` directory. The default value is `deployedResources`.

com.tivoli.pd.as.rbpf.ResourceContainerName=Resources:

This property specifies the Tivoli Access Manager object space container name for the protected resources. The default location is the `/WebAppServer/deployedResources/Resources` directory.

com.tivoli.pd.as.rbpf.RoleContainerName=Roles:

This property specifies the Tivoli Access Manager protected object space container name for the security roles. The default location is the `/WebAppServer/deployedResources/Roles` directory.

System-dependent configuration properties:

Do not change these system-dependent configuration properties. These properties are included in this article for reference only.

These properties are in the *profile_root*`/etc/tam/amwas.`*node_name_server_name*`.amjacc.properties` file.

The *profile_root* variable is the value of the profilePath parameter when the profile is created.

com.tivoli.pd.as.rbpf.AmasSession.CfgURL=file/:$WAS_HOME/profiles/profile_name/etc/tam/amwas.node_server.pdperm.properties:

This entry is generated by the Java Authorization Contract for Containers (JACC) provider configuration. This argument specifies the location of the file that contains information about the JACC provider of Tivoli Access Manager. Do not change this entry or the properties in the amwas.node_server.pdperm.properties file.

com.tivoli.pd.as.rbpf.AmasSession.CfgURL=file/:user_root/etc/tam/amwas.node_server.pdperm.properties:

This entry is generated by the Java Authorization Contract for Containers (JACC) provider configuration. It specifies the location of the file that contains information about the Tivoli Access Manager JACC provider. Do not change this entry or the properties in the amwas.node_server.pdperm.properties file.

com.tivoli.pd.as.rbpf.AmasSession.LoggingURL=file/:$WAS_HOME/profiles/profile_name/etc/tam/amwas.node_server.pdjlog.properties:

This entry contains the location of the logging configuration file for the JACC provider of Tivoli Access Manager. The referenced file is generated by the JACC provider of Tivoli Access Manager configuration. Do not change this entry.

com.tivoli.pd.as.rbpf.AmasSession.LoggingURL=file/:user_root/etc/tam/amwas.node_server.pdjlog.properties:

This entry contains the location of the logging configuration file for the Tivoli Access Manager JACC provider. The file referenced is generated by the Tivoli Access Manager JACC provider configuration. Do not change this entry.

Administering security users and roles with Tivoli Access Manager:

Use these steps to manage user-to-role mappings and user-to-group mappings for applications.

About this task

User-to-role mapping and user-to-group mapping for the JACC provider of Tivoli Access Manager are performed using the WebSphere Application Server administrative console.

Procedure

1. Click **Applications > Enterprise applications >** *application_name*.
2. Under Additional properties, click **Security role to user/group mapping**. The user and groups management screen is displayed.
3. Select the role that requires user or group management and use **Lookup users** or **Lookup groups** to manage the users or groups for the selected role. The native role mapping uses the MapRolesToUsers administrative task. If you are using Tivoli Access Manager, use the TAMMapRolesToUsers administrative task instead. The syntax and options for the Tivoli version are the same as those used in the native version. For more information, see “Role-based security with embedded Tivoli Access Manager” on page 1633 and “Configuring Tivoli Access Manager groups.”

Configuring Tivoli Access Manager groups:

Use these steps to configure the WebSphere Application Server administrative console to add objects of the accessGroup class to the list of object classes that represent user registry groups.

About this task

You can use the WebSphere Application Server administrative console to specify security policies for applications that run in the WebSphere Application Server environment. You can also use the WebSphere Application Server administrative console to specify security policies for other web resources, based on the entities that are stored in the user registry.

Tivoli Access Manager adds the accessGroup object class to the registry. Tivoli Access Manager administrators can use the pdadmin utility, which is available only on the policy server host in the PD.RTE fileset, to create new groups. These new groups are added to the registry as the accessGroup object class.

The WebSphere Application Server administrative console is not configured by default to recognize objects of the accessGroup class as user registry groups. You can configure the WebSphere Application Server administrative console to add this object class to the list of object classes that represent user registry groups. To do this configuration, complete the following instructions:

Procedure

1. From the WebSphere Application Server administrative console, access the advanced settings for configuring security by clicking **Security > Global security**.
2. Under User account repository, click the **Available realm definitions** drop-down list, select **Standalone LDAP registry**, and click **Configure**.
3. Under Additional properties, click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.
4. Modify the **Group Filter** field. Add the following entry: `(objectclass=accessGroup)`

   The Group Filter field looks like the following example:

   ```
   (&(cn=%w)(|(objectclass=groupOfNames)
   (objectclass=groupOfUniqueNames)(objectclass=accessGroup)))
   ```
5. Modify the **Group Member ID Map** field. Add the following entry: `accessGroup:member`. The Group Member ID Map field looks like the following example:

   ```
   groupOfNames:member;groupOfUniqueNames:uniqueMember;
   accessGroup:member
   ```
6. Stop and restart WebSphere Application Server.

Configuring additional authorization servers for Tivoli Access Manager:

Tivoli Access Manager secure domains can contain more than one authorization server. Having multiple authorization servers is useful for providing a failover capability as well as improving performance when the volume of access requests is large.

Procedure

1. Refer to the *Tivoli Access Manager Base Administration Guide* for details on installing and configuring authorization servers. This document is available in the IBM Tivoli Access Manager for e-business information center.
2. Re-configure the Java Authorization Contract for Containers (JACC) provider using the `$AdminTask reconfigureTAM interactive` **wsadmin** command. Enter all new and existing options. The following table lists the information that you are asked to provide for the reconfigureTAM command. The table also lists the properties that apply to the configureTAM and unconfigureTAM commands.

Table 133. Commands for configuring, reconfiguring, and unconfiguring Tivoli Access Manager. The following table lists the information that you are asked to provide for the configureTAM command. The table also lists the properties that apply to the unconfigureTAM and reconfigureTAM commands.

Property	Default	Relevant command	Description
Websphere Application Server node name	*	• configureTAM • reconfigureTAM • unconfigureTAM	Specify a single node on which to run the configuration task.
Tivoli Access Manager Policy Server	Default port: 7135	• configureTAM • reconfigureTAM	Enter the name of the Tivoli Access Manager policy server and the connection port. Use the format, *policy_server* **:** *port*. The policy server communication port is set at the time of Tivoli Access Manager configuration.
Tivoli Access Manager Authorization Server	Default port: 7136	• configureTAM • reconfigureTAM	Enter the name, port, and priority of each configured Tivoli Access Manager authorization server. Use the format *auth_server* **:** *port* **:** *priority*. The authorization server communication port is set at the time of Tivoli Access Manager configuration. You can specify more than one authorization server by separating the entries with commas. Having more than one authorization server configured is useful for failover and performance. The priority value is the order of authorization server use. For example: auth_server1**:**7136**:**1,auth_server2**:**7137**:**2. A priority of 1 is still required when you use a single authorization server.
Websphere Application Server administrator's distinguished name		• configureTAM • reconfigureTAM	Enter the full distinguished name of the security primary administrator ID for WebSphere Application Server as created in the "Creating the security administrative user" topic in the *Securing applications and their environment* PDF. For example: cn=wasadmin,o=*organization*,c=*country*
Tivoli Access Manager user registry distinguished name suffix		• configureTAM • reconfigureTAM	Enter the suffix that you have set up in the user registry to contain the user and groups for Tivoli Access Manager. For example: o=*organization*,c=*country*

Table 133. Commands for configuring, reconfiguring, and unconfiguring Tivoli Access Manager (continued). The following table lists the information that you are asked to provide for the configureTAM command. The table also lists the properties that apply to the unconfigureTAM and reconfigureTAM commands.

Property	Default	Relevant command	Description
Tivoli Access Manager administrator's user name	sec_master	• configureTAM • reconfigureTAM • unconfigureTAM	Enter the Tivoli Access Manager administration user ID that you created when you configured Tivoli Access Manager. This ID is usually sec_master.
Tivoli Access Manager administrator's user password		• configureTAM • reconfigureTAM • unconfigureTAM	Enter the password that is associated with the Tivoli Access Manager administration user ID.
Tivoli Access Manager security domain	Default	• configureTAM • reconfigureTAM	Enter the name of the Tivoli Access Manager security domain that is used to store users and groups. If a security domain is not already established at the time of Tivoli Access Manager configuration, click **Return** to accept the default.
Embedded Tivoli Access Manager listening port set	8900:8999	• configureTAM • reconfigureTAM	WebSphere Application Server needs to listen on a TCP/IP port for authorization database updates from the policy server. More than one process can run on a particular node and machine so a list of ports is required for the processes. Enter the ports that are used as listening ports by Tivoli Access Manager clients, separated by a comma. If you specify a range of ports, separate the lower and higher values by a colon. For example, 7999, 9990:9999.
Defer	No	• configureTAM • reconfigureTAM • unconfigureTAM	Set this option to *yes* if you want to defer the configuration of the management server until the next restart. Set the option to *no* if you want the configuration of the management server to occur immediately. Managed servers are configured on their next restart.
Force	No	• reconfigureTAM • unconfigureTAM	Set this value to *yes* if you want to ignore errors during the unconfiguration process and allow the entire process to complete. Set the value to *no* if you want errors to stop the unconfiguration process. This option is especially useful if the environment needs to be cleaned up and problems are occurring that do not allow the entire cleanup process to complete successfully.

Logging Tivoli Access Manager security:

Use this topic to enable the trace specification to indicate tracing at the required level.

About this task

The Java Authorization Contract for Containers (JACC) for Tivoli Access Manager provider messages are logged to the configured trace output location, and messages are written to standard out `SystemOut.log` file. When trace is enabled, all logging, both trace and messaging, is sent to the `trace.log` file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Procedure

1. The `amwas.`*`node_server`*`.pdjlog.properties` file must be updated and the **isLogging** attribute set to *true* for the required component. For example, to enable tracing for the JACC provider for Tivoli Access Manager, set the following line to *true*:
   `amwas.`*`node_server`*`.pdjlog.properties:baseGroup.AMWASWebTraceLogger.isLogging=true`
2. Enable tracing for the JACC provider of Tivoli Access Manager components in the WebSphere Application Server administrative console by completing the following steps:
   a. Click **Troubleshooting** > **Logs and Trace** > *server_name*.
   b. Under Logs and Trace tasks, click **Diagnostic trace**.
   c. Select the **Enable Log** option.
   d. Click **Apply**.
   e. Click **Troubleshooting** > **Logs and Trace** > *server_name*.

f. Click **Change Log Detail Levels**.

g. Click **Components**. Tracing for all components can be enabled using the **com.tivoli.pd.as.*** command. Tracing for separate components can be enabled using the following commands:
   - **com.tivoli.pd.as.rbpf.*** for role-based policy framework tracing
   - **com.tivoli.pd.as.jacc.*** for JACC provider tracing
   - **com.tivoli.pd.as.pdwas.*** for the authorization table
   - **com.tivoli.pd.as.cfg.*** for configuration
   - **com.tivoli.pd.as.cache.*** for caching

   For more information, see ../ae/utrb_loglevel.dita.

h. Click **Apply**.

What to do next

The trace specification now indicates that tracing is enabled at the required level. Save the configuration and restart the server for the changes to take effect.

Tivoli Access Manager loggers:

The Java Authorization Contract for Containers (JACC) provider for Tivoli Access Manager uses the JLog logging framework as does the Java runtime environment for Tivoli Access Manager. You can enable tracing and messaging selectively for specific JACC providers for Tivoli Access Manager components.

The JACC for Tivoli Access Manager provider messages are logged to the configured trace output location, and messages are written to standard out `SystemOut.log` file. When trace is enabled, all logging, both trace and messaging, is sent to the `trace.log` file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Tracing and message logging for the JACC provider for Tivoli Access Manager are configured in the `amwas.`*`node_server`*`.pdjlog.properties` properties file, which is located in the `profile_root/etc/tam` directory. This file contains logging properties from the `amwas.pdjlog.template.properties` template file for the specific node and server combination at the time of JACC provider for Tivoli Access Manager configuration.

The contents of this file let the user control:

- Whether tracing is enabled or disabled for the JACC provider of Tivoli Access Manager components.
- Whether message logging is enabled or disabled for the JACC provider of Tivoli Access Manager components.

The `amwas.`*`node_server`*`.pdjlog.properties` file defines several loggers, each of which is associated with one JACC provider of Tivoli Access Manager component. These loggers include:

Table 134. Tivoli Access Manager loggers. This table describes the Tivoli Access Manager loggers.

Logger Name	Description
AmasRBPFTraceLogger AmasRBPFMessageLogger	Logs messages and trace for the role-based policy framework. This underlying framework is used by embedded Tivoli Access Manager to make access decisions.
AmasCacheTraceLogger AmasCacheMessageLogger	Logs messages and trace for the policy caches that are used by the role-based policy framework.

Table 134. Tivoli Access Manager loggers (continued). This table describes the Tivoli Access Manager loggers.

Logger Name	Description
AMWASWebTraceLogger AMWASWebMessageLogger	Logs messages and trace for the WebSphere Application Server authorization plug-in.
AMWASConfigTraceLogger AMWASConfigMessageLogger	Logs messages and trace for the configuration actions of the JACC provider for Tivoli Access Manager .
JACCTraceLogger JACCMessageLogger	Logs messages and trace for the JACC provider activity of Tivoli Access Manager .

Note: Tracing can have a significant impact on system performance. Enable tracing only when diagnosing the cause of a problem.

The implementation of these loggers routes messages to the WebSphere Application Server logging sub-system. All messages are written to the WebSphere Application Server `trace.log` file.

For each logger, the `amwas.`*`node_server`*`.pdjlog.properties` file defines an isLogging attribute which, when set to *true*, enables logging for the specific component. A value of *false* disables logging for that component.

The `amwas.node_server.pdjlog.properties` file defines the parent loggers MessageLogger and TraceLogger that also have an isLogging attribute. If the child loggers do not specify this isLogging attribute, they inherit the value of their respective parent. When the JACC provider for Tivoli Access Manager is enabled, the isLogging attribute is set to *true* for the MessageLogger and set to *false* for the TraceLogger logger. Message logging is enabled for all components and tracing is disabled for all components, by default.

To turn on tracing for a JACC provider component, see Logging Tivoli Access Manager security.

Interfaces that support JACC:

WebSphere Application Server provides the RoleConfigurationFactory and the RoleConfiguration interfaces, which are similar to PolicyConfigurationFactory and PolicyConfiguration interfaces so the information that is stored in the bindings file can be propagated to the provider during installation. The implementation of these interfaces is optional.

RoleConfiguration interface

Use the RoleConfiguration interface to propagate the authorization information to the provider. This interface is similar to the PolicyConfiguration interface that is found in Java Authorization Contact for Containers (JACC).

```
RoleConfiguration
      - com.ibm.wsspi.security.authorization.RoleConfiguration

/**
 * This interface is used to propagate the authorization table information
 * in the binding file during application installation. Implementation of this interface is
 * optional. When a JACC provider implements this interface during an application, both
 * the policy and the authorization table information are propagated to the provider.
* If this is not implemented, only the policy information is propagated as per
 * the JACC specification.
 * @ibm-spi
 * @ibm-support-class-A1
 */


public interface RoleConfiguration

/**
 * Add the users to the role in RoleConfiguration.
 * The role is created, if it does not exist in RoleConfiguration.
 * @param role the role name.
 * @param users the list of the user names.
 * @exception RoleConfigurationException if the users cannot be added.
 */
 public void addUsersToRole(String role, List users)
 throws RoleConfigurationException
```

```
/**
 * Remove the users to the role in RoleConfiguration.
 * @param role the role name.
 * @param users the list of the user names.
 * @exception RoleConfigurationException if the users cannot be removed.
 */
 public void removeUsersFromRole(String role, List users)
 throws RoleConfigurationException

/**
 * Add the groups to the role in RoleConfiguration.
 * The role is created if it does not exist in RoleConfiguration.
 * @param role the role name.
 * @param groups the list of the group names.
 * @exception RoleConfigurationException if the groups cannot be added.
 */
 public void addGroupsToRole(String role, List groups)
 throws RoleConfigurationException

/**
 * Remove the groups to the role in RoleConfiguration.
 * @param role the role name.
 * @param groups the list of the group names.
 * @exception RoleConfigurationException if the groups cannot be removed.
 */
 public void removeGroupsFromRole( String role, List groups)
 throws RoleConfigurationException

/**
 * Add the everyone to the role in RoleConfiguration.
 * The role is created if it does not exist in RoleConfiguration.
 * @param role the role name.
 * @exception RoleConfigurationException if the everyone cannot be added.
 */
 public void addEveryoneToRole(String role)
 throws RoleConfigurationException

/**
 * Remove the everyone to the role in RoleConfiguration.
 * @param role the role name.
 * @exception RoleConfigurationException if the everyone cannot be removed.
 */
 public void removeEveryoneFromRole( String role)
 throws RoleConfigurationException

/**
 * Add the all authenticated users to the role in RoleConfiguration.
 * The role is created if it does not exist in RoleConfiguration.
 * @param role the role name.
 * @exception RoleConfigurationException if the authentication users cannot
 *  be added.
 */
 public void addAuthenticatedUsersToRole(String role)
 throws RoleConfigurationException

/**
 * Remove the all authenticated users to the role in RoleConfiguration.
 * @param role the role name.
 * @exception RoleConfigurationException if the authentication users cannot
 * be removed.
 */
 public void removeAuthenticatedUsersFromRole( String role)
 throws RoleConfigurationException

/**
 * This commits the changes in Roleconfiguration.
 * @exception RoleConfigurationException if the changes cannot be
 * committed.
 */
 public void commit( )
 throws RoleConfigurationException

/**
 * This deletes the RoleConfiguration from the RoleConfiguration Factory.
 * @exception RoleConfigurationException if the RoleConfiguration cannot
 * be deleted.
 */
 public void delete( )
 throws RoleConfigurationException

/**
 * This returns the contextID of the RoleConfiguration.
 * @exception RoleConfigurationException if the contextID cannot be
 * obtained.
 */
 public String getContextID( )
 throws RoleConfigurationException
```

RoleConfigurationFactory interface

The RoleConfigurationFactory interface is similar to the PolicyConfigurationFactory interface that is introduced by JACC, and is used to obtain RoleConfiguration objects based on the contextID IDs.

```
RoleConfigurationFactory
 - com.ibm.wsspi.security.authorization.RoleConfigurationFactory

/**
 * This interface is used to instantiate the com.ibm.wsspi.security.authorization.RoleConfiguration
 * objects based on the context identifier similar to the policy context identifier.
 * Implementation of this interface is required only if the RoleConfiguration interface is implemented.
 *
 * @ibm-spi
 * @ibm-support-class-A1
 */

public interface RoleConfigurationFactory
/**
 * This gets a RoleConfiguration with contextID from the
 * RoleConfigurationfactory. If the RoleConfiguration does not exist
 * for the contextID in the RoleConfigurationFactory, a new
 * RoleConfiguration with contextID is created in the
 * RoleConfigurationFactory. The contextID is similar to
 * PolicyContextID, but it does not contain the module name.
 * If remove is true, the old RoleConfiguration is removed and a new
 * RoleConfiguration is created, and returns with the contextID.
 * @return the RoleConfiguration object for this contextID
 * @param contextID the context ID of RoleConfiguration
 * @param remove true or false
 * @exception RoleConfigurationException if RoleConfiguration
 * cannot be obtained.
 **/
public abstract com.ibm.ws.security.policy.RoleConfiguration
         getRoleConfiguration(String contextID, boolean remove)
      throws RoleConfigurationException
```

InitializeJACCProvider provider

When implemented by the provider, this interface is called by every process where the JACC provider can be used for authorization. All additional properties that are entered during the authorization check are passed to the provider. For example, the provider can use this information to initialize client code to communicate with their server or repository. The cleanup method is called during server shutdown to clean up the configuration.

Declaration

public interface **InitializeJACCProvider**

Description

This interface has two methods. The JACC provider can implement the interface, and WebSphere Application Server calls it to initialize the JACC provider. The name of the implementation class is obtained from the value of the initializeJACCProviderClassName system property.

This class must reside in a Java archive (JAR) file on the class path of each server that uses this provider.

```
InitializeJACCProvider
   - com.ibm.wsspi.security.authorization.InitializeJACCProvider

  /**
   * Initializes the JACC provider
      * @return 0 for success.
   * @param props the custom properties that are included for this provider will
   * pass to the implementation class.
   * @exception Exception for any problems encountered.
   **/
  public int initialize(java.util.Properties props)
  throws Exception

  /**
   * This method is for the JACC provider cleanup and will be called during a process stop.
   **/
  public void cleanup()
```

Enabling the JACC provider for Tivoli Access Manager:

The Java Authorization Contract for Container (JACC) provider for Tivoli Access Manager is configured by default. Use this topic to enable the JACC provider for Tivoli Access Manager.

About this task

Restriction: Do not perform this task if you are configuring the JACC provider for Tivoli Access Manager to supply authentication services only. Only perform this task for installations that require both Tivoli Access Manager authentication and authorization protection.

The JACC provider for Tivoli Access Manager is configured by default. To enable the JACC provider for Tivoli Access Manager, complete the following steps:

Procedure

1. Click **Security > Global security > External authorization providers**.
2. Select the **External authorization using a JACC provider** option, then click **Apply**.
3. Under Related Items, click **External JACC provider**. The JACC provider settings for Tivoli Access Manager are displayed.
4. Verify that the correct settings are present to work with your Tivoli Access Manager configuration. The following list shows the JACC provider configuration settings for Tivoli Access Manager.

Table 135. JACC provider configuration settings for Tivoli Access Manager. This table describes the JACC provider configuration settings for Tivoli Access Manager.

Field	Value
Name	Tivoli Access Manager
Description	This field is optional and used as a reference.
J2EE policy class name	`com.tivoli.pd.as.jacc.TAMPolicy`
Policy configuration factory class name	`com.tivoli.pd.as.jacc.TAMPolicyConfigurationFactory`
Role configuration factory class name	`com.tivoli.pd.as.jacc.TAMRoleConfigurationFactory`
JACC provider initialization class name	`com.tivoli.pd.as.jacc.cfg.TAMConfigInitialize`
Requires the EJB arguments policy context handler for access decisions	false
Supports dynamic module updates	true

For more information, see “External Java Authorization Contract for Containers provider settings” on page 1646.

5. Under Additional properties, click **Tivoli Access Manager properties** and set the properties that are associated with the embedded Tivoli Access Manager. The following table explains the properties that are needed for the embedded Tivoli Access Manager. Some fields do not have default values.

Table 136. Tivoli Access Manger properties. This table lists the Tivoli Access Manger properties.

Name	Default value	Description
Enable embedded Tivoli Access Manager	Unchecked	When you select this check box, the embedded Tivoli Access Manager is configured or reconfigured. When you clear this check box, the embedded Tivoli Access Manager is unconfigured.
Ignore errors during embedded Tivoli Access Manager disablement	Unchecked	If you check this check box and click **OK** or **Apply**, when you unconfigure the embedded Tivoli Access Manager, any unconfiguration errors are ignored and the process completes. If you do not check this check box, unconfiguration errors cause the unconfiguration process to stop.
Client listening port	8900:8999	When the embedded Tivoli Access Manager is configured and running, it requires several ports to listen for updates to the access control list database for Tivoli Access Manager. The value in this field is a range of port numbers that Tivoli Access Manager can use for this purpose. The first 20% of this range is reserved for the deployment manager. You can enter multiple ranges or individual port numbers in a line separated list. For example: 8900:8999 9100:9200 9999

Table 136. Tivoli Access Manger properties (continued). This table lists the Tivoli Access Manger properties.

Name	Default value	Description
Policy server		This field value specifies the name and port number of the configure and running Tivoli Access Manager policy server. The format is server:port For example:`snapper.ibm.com:7135`
Authorization servers		This field contains the names, port numbers, and priorities of all of the configured and running Tivoli Access Manager authorization servers. This field must contain at least one authorization server. If multiple authorization servers are listed, those servers are used for failover. The server with priority 1 is used first with failover to server priority 2 and so on. The format is server:port:priority with each authorization server listed on a different line. For example: snapper.ibm.com:7136:1 turtle.ibm.com:7136:2
Authorization user name	sec_master	This field value specifies the administrative user name for Tivoli Access Manager.
Administrator user password		This field value specifies the password for Tivoli Access Manager.
User registry distinguished name suffix		This field value is the suffix that is set up in the user registry to contain the users and groups for Tivoli Access Manager. For example using IBM Tivoli Directory Server: o=ibm,c=au
Security domain	Default	This field value specifies the configured security domain to use for the embedded Tivoli Access Manager.
Administrator user distinguished name		This field specifies the fully distinguished user name of the primary administrative user for WebSphere Application Server security. For example using IBM Tivoli Directory Server: cn=wasadmin,o=ibm,c=au

For more information, see “Tivoli Access Manager JACC provider settings” on page 1652.

6. Click **OK**.
7. Save the settings by clicking **Save** at the top of the page.
8. Log out of the WebSphere Application Server administrative console.
9. Restart WebSphere Application Server. The security configuration is now replicated to managed servers and node agents. These other servers within a cell also require restarting before the security changes take effect.

Enabling embedded Tivoli Access Manager:

Embedded Tivoli Access Manager is not enabled by default, and you need to configure it for use.

About this task

Enabling Tivoli Access Manager security within WebSphere Application Server requires:

- A supported Lightweight Directory Access Protocol (LDAP) installed somewhere on your network. This user registry contains the user and group information for both Tivoli Access Manager and WebSphere Application Server.
- Tivoli Access Manager server exists and is configured to use the user registry. For details on the installation and configuration of Tivoli Access Manager, refer to the IBM Tivoli Access Manager for e-business information center.

Note: WebSphere Application Server contains an embedded client for Tivoli Access Manager. To use Tivoli Access Manager, you must also configure the Tivoli Access Manager server.

However, the server version must be the same version or later as the client version. For information on the supported version of Tivoli Access Manager, see WebSphere Application Server - Supported Prerequisites.

- WebSphere Application Server is installed either in a single server model or as WebSphere Application Server, Network Deployment.
- When administrative security is configured with a Federal Information Processing Standard (FIPS) provider, the Tivoli Access Manager server must be configured for FIPS as well

Complete the following steps to enable embedded Tivoli Access Manager security:

Procedure

1. Create the security administrative user.
   For more information, see the *Securing applications and their environment* PDF.
2. Configure the Java Authorization Contract for Containers (JACC) provider for Tivoli Access Manager .
   For more information, see the *Securing applications and their environment* PDF.
3. Enable WebSphere Application Server security. When you are using Tivoli Access Manager you must configure LDAP as the user registry.
   For more information, see the *Securing applications and their environment* PDF.
4. Enable the JACC provider for Tivoli Access Manager.
   For more information, see the *Securing applications and their environment* PDF.

TAMConfig command group for the AdminTask object:

You can use the Jython or Jacl scripting languages to configure embedded IBM Tivoli Access Manager with the wsadmin tool. The commands and parameters in the TAMConfig group can be used to configure or unconfigure Tivoli Access Manager.

The TAMConfig command group for the AdminTask object includes the following commands:

- "configureTAM"
- "listTAMSettings" on page 1668
- "modifyTAM" on page 1668
- "reconfigureTAM" on page 1669
- "unconfigureTAM" on page 1669
- "configureTAMTAI" on page 1670
- "unconfigureTAMTAI" on page 1672
- "configureTAMTAIProperties" on page 1673
- "unconfigureTAMTAIProperties" on page 1675
- "configureTAMTAIPdjrte" on page 1675
- "unconfigureTAMTAIPdjrte" on page 1677

configureTAM

Use the configureTAM command to manually configure the Tivoli Access Manager.

Target object

None.

Required parameters

None.

Optional parameters

None.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configureTAM {-interactive}
```

- Using Jython:

```
AdminTask.configureTAM('-interactive')
```

listTAMSettings

The listSSLRepertoires command displays the current embedded Tivoli Access Manager configuration settings.

Target object

None.

Required parameters

None.

Optional parameters

None.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask listTAMSettings {-interactive}
```

- Using Jython:

```
print AdminTask.listTAMSettings('-interactive')
```

modifyTAM

The modifyTAM command modifies embedded Tivoli Access Manager configuration settings.

Target object

None.

Required parameters

-adminPasswd
Specifies the Tivoli Access Manager administrator password. (String, required)

Optional parameters

-adminUid
Specifies the Tivoli Access Manager user name. (String, optional)

-nodeName
Specifies the target node or nodes. Set the value as the * asterisk character to specify all nodes. (String, optional)

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask modifyTAM {-adminPasswd my11password}
```

- Using Jython:

```
AdminTask.modifyTAM('-adminPasswd my11password')
```

- Using Jython list:

```
AdminTask.modifyTAM(['-adminPasswd', 'my11password'])
```

Interactive mode example usage:

- Using Jacl:

```
$AdminTask modifyTAM {-interactive}
```

- Using Jython:

```
AdminTask.modifyTAM('-interactive')
```

reconfigureTAM

The reconfigureTAM command reconfigures the Java Authorization Contract for Containers (JACC) Tivoli Access Manager settings.

Target object

None.

Required parameters

None.

Optional parameters

None.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask reconfigureTAM {-interactive}
```

- Using Jython:

```
AdminTask.reconfigureTAM('-interactive')
```

unconfigureTAM

The unconfigureTAM command removes configuration data for the Java Authorization Contract for Containers (JACC) Tivoli Access Manager.

Required parameters

None.

Optional parameters

None.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask unconfigureTAM {-interactive}
```

- Using Jython:

```
AdminTask.unconfigureTAM('-interactive')
```

configureTAMTAI

The configureTAMTAI command configures the embedded Tivoli Access Manager trust association interceptor (TAI) with classname TAMTrustAsociationInterceptorPlus.

Target object

None.

Required parameters

-policySvr
This property specifies the name of the Tivoli Access Manager policy server with which the application server communicates. The server is specified by a fully-qualified host name, the SSL port number, and the rank. The default SSL port number is 7135. For example: `myauth.mycompany.com:7135:1`.

-authSvrs
This property specifies the name of the Tivoli Access Manager authorization server with which the application server communicates. The server is specified by a fully-qualified host name, the SSL port number, and the rank. The default SSL port number is 7136. For example: `myauth.mycompany.com:7136:1`. You can specify multiple servers if the entries are separated by a comma (,).

-adminPasswd
This property specifies the password of the Tivoli Access Manager administrator user that is associated with the -adminUid parameter. The password restrictions depend upon the password policy for your Tivoli Access Manager configuration.

-loginId
The WebSEAL trusted user as created in "Creating a trusted user account in Tivoli Access Manager". See the Configuring single sign-on using trust association interceptor ++ article for more information. The format of the username is the short name representation.

Optional parameters

-adminUid
This property specifies the Tivoli Access Manager administrator name. If this option is not specified, `sec_master` is the default. A valid administrative ID is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the administrative ID.

For example, for U.S. English, the valid characters are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). The minimum and maximum lengths of the administrative ID, if there are limits, are imposed by the underlying registry.

-secDomain
This property specifies the Tivoli Access Manager domain name to which the administrator is authenticated. This domain must exist and an administrator ID and password must be valid for this domain. The application server is specified in this domain. If the application server is not specified, the default value is `Default`. The local domain value is retrieved from the configuration file.

A valid domain name is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the domain name.

For example, for U.S. English, the valid characters for domain names are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). The minimum and maximum lengths of the domain name, if there are limits, are imposed by the underlying registry.

-checkViaHeader

You can configure TAI so that the via header can be ignored when validating trust for a request. Set this property to `false` if none of the hosts in the via header need to be trusted. When set to false, you do not need to set the trusted host names and host ports properties. The only mandatory property to check when the via header is false is com.ibm.websphere.security.webseal.loginId. The default value of the check via header property is `false`. When using Tivoli Access Manager plug-in for web servers, set this property to false.

Note: The via header is part of the standard HTTP header that records the server names that the request passed through.

-id

This property specifies a comma-separated list of headers that exists in the request. If all of the configured headers do not exist in the request, trust cannot be established. The default value for the ID property is `iv-creds`. Any other values set in WebSphere Application Server are added to the list along with iv-creds, separated by commas.

-hostnames

Do not set this property if you are using the Tivoli Access Manager plug-in for web servers. This property specifies the host names (case-sensitive) that are both trusted and expected in the request header. Requests arriving from unlisted hosts might not be trusted. If the checkViaHeader property is not set, or is set to `false`, then the trusted host names property has no influence. If the checkViaHeader property is set to `true`, and the trusted host names property is not set, the TAI initialization fails.

-ports

Do not set this property if you are using the Tivoli Access Manager plug-in for web servers. This property is a comma-separated list of trusted host ports. Requests that arrive from unlisted ports might not be trusted. If the checkViaHeader property is not set, or is set to `false`, then this property has no influence. If the checkViaHeader property is set to `true`, and the trusted host ports property is not set in WebSphere Application Server, the TAI initialization fails.

-viaDepth

This property indicates a positive integer that specifies the number of source hosts in the via header to check for trust. By default, every host in the via header is checked, and if any host is not trusted, trust cannot be established. The viaDepth property is used when only some of the hosts in the via header have to be trusted. The setting indicates the number of hosts that are required to be trusted.

For example, consider the following header:

If in via: HTTP/1.1 webseal1:7002, 1.1 webseal2:7001If the viaDepth property is not set, is set to 2 or is set to 0, and a request with the previous via header is received then both webseal1:7002 and webseal2:7001 need to be trusted. The following configuration then applies:

```
com.ibm.websphere.security.webseal.hostnames = webseal1,webseal2
```

If in com.ibm.websphere.security.webseal.ports = 7002,7001If the viaDepth property is set to 1, and the previous request is received, then only the last host in the via header needs to be trusted. The following configuration then applies:

```
com.ibm.websphere.security.webseal.hostnames = webseal2
com.ibm.websphere.security.webseal.ports = 7001
```

The viaDepth property is set to 0 by default, which means that all of the hosts in the via header are checked for trust.

-ssoPwdExpiry
After trust is established for a request, the single sign-on user password is cached, eliminating the need to have the TAI re-authenticate the single sign-on user with Tivoli Access Manager for every request. You can modify the cache timeout period by setting the single sign-on password expiry property to the required time in seconds. If the password expiry property is set to 0, the cached password never expires. The default value for the password expiry property is 600.

-ignoreProxy
This property can be used to tell the TAI to ignore proxies as trusted hosts. If set to `true` the comments field of the hosts entry in the via header is checked to determine if a host is a proxy. Remember that not all proxies insert comments in the via header indicating that they are proxies. The default value of the ignoreProxy property is `false`. If the checkViaHeader property is set to false, then the ignoreProxy property has no influence in establishing trust.

-configURL
For the TAI to establish trust for a request, it requires that the SvrSslCfg task be run for the Java Virtual Machine on the Application Server and result in the creation of a properties file. If this properties file is not at the default URL, which is file://java.home/PdPerm.properties, the correct URL of the properties file must be set in the configuration URL property. If this property is not set, and the SvrSslCfg-generated properties file is not in the default location, the TAI initialization fails. The default value for the config URL property is `file://${WAS_INSTALL_ROOT}/java/jre/PdPerm.properties.`

-defer
This property indicates whether the Tivoli Access Manager configuration portion of this task should be run immediately or deferred until the startup of the WebSphere Application Server. The default value is `no`.

Note: The TAI properties are updated immediately regardless of this setting.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configureTAMTAI {-interactive}
```

- Using Jython:

```
AdminTask.configureTAMTAI('-interactive')
```

unconfigureTAMTAI

The unconfigureTAMTAI command unconfigures the embedded Tivoli Access Manager Trust Association Interceptor with classname TAMTrustAsociationInterceptorPlus. This task does not include removing any custom properties from the security configuration.

Target object

None.

Required parameters

-adminPasswd
Specifies the password of the Tivoli Access Manager administrator user that is associated with the -adminUid parameter. The password restrictions depend upon the password policy for your Tivoli Access Manager configuration.

Optional parameters

-adminUid
Specifies the Tivoli Access Manager administrator name. If this option is not specified, `sec_master` is

the default. A valid administrative ID is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the administrative ID.

For example, for U.S. English the valid characters are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). The minimum and maximum lengths of the administrative ID, if there are limits, are imposed by the underlying registry.

-force
Indicates whether or not this task should stop when an error is encountered. The default value is no.

-defer
Indicates whether this task should be run immediately or deferred until the startup of the WebSphere Application Server. The default value is no.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask unconfigureTAMTAI {-interactive}
```

- Using Jython:

```
AdminTask.unconfigureTAMTAI('-interactive')
```

configureTAMTAIProperties

The configureTAMTAIProperties command adds the custom properties to the security configuration for the embedded Tivoli Access Manager Trust Association Interceptor with classname TAMTrustAsociationInterceptorPlus.

Target object

None.

Required parameters

-loginId
The WebSEAL trusted user is created as discussed in "Creating a trusted user account in Tivoli Access Manager". See the Configuring single sign-on using trust association interceptor ++ article for more information. The format of the username is the short name representation.

Optional parameters

-checkViaHeader
You can configure TAI so that the via header can be ignored when validating trust for a request. Set this property to false if none of the hosts in the via header need to be trusted. When set to `false` you do not need to set the trusted host names and host ports properties. The only mandatory property to check when via header is false is com.ibm.websphere.security.webseal.loginId. The default value of the check via header property is `false`. When using Tivoli Access Manager plug-in for web servers, set this property to false.

Note: The via header is part of the standard HTTP header that records the server names that the request passed through.

-id
This property indicates a comma-separated list of headers that exists in the request. If all of the

configured headers do not exist in the request, trust cannot be established. The default value for the ID property is `iv-creds`. Any other values set in WebSphere Application Server are added to the list along with iv-creds, separated by commas.

-hostnames
Do not set this property if using Tivoli Access Manager plug-in for web servers. The property specifies the host names (case-sensitive) that are both trusted and expected in the request header. Requests arriving from unlisted hosts might not be trusted. If the checkViaHeader property is not set, or is set to `false`, then the trusted host names property has no influence. If the checkViaHeader property is set to `true`, and the trusted host names property is not set, the TAI initialization fails.

-ports
Do not set this property if you are using the Tivoli Access Manager plug-in for web servers. This property is a comma-separated list of trusted host ports. Requests that arrive from unlisted ports might not be trusted. If the checkViaHeader property is not set, or is set to `false`, then this property has no influence. If the checkViaHeader property is set to `true`, and the trusted host ports property is not set in WebSphere Application Server, the TAI initialization fails.

-viaDepth
This property indicates a positive integer that specifies the number of source hosts in the via header to check for trust. By default, every host in the via header is checked, and if any host is not trusted, trust cannot be established. The viaDepth property is used only when some of the hosts in the via header have to be trusted. The setting indicates the number of hosts that are required to be trusted.

As an example, consider the following header:

If in via: HTTP/1.1 webseal1:7002, 1.1 webseal2:7001If the viaDepth property is not set, is set to 2 or is set to 0, and a request with the previous via header is received then both webseal1:7002 and webseal2:7001 need to be trusted. The following configuration then applies:

```
com.ibm.websphere.security.webseal.hostnames = webseal1,webseal2
```

If in com.ibm.websphere.security.webseal.ports = 7002,7001If the viaDepth property is set to 1, and the previous request is received, then only the last host in the via header needs to be trusted. The following configuration then applies:

```
com.ibm.websphere.security.webseal.hostnames = webseal2
com.ibm.websphere.security.webseal.ports = 7001
```

The viaDepth property is set to 0 by default, which means that all of the hosts in the via header are checked for trust.

-ssoPwdExpiry
This property can be used to tell the TAI to ignore proxies as trusted hosts. If set to true, the comments field of the hosts entry in the via header is checked to determine if a host is a proxy. Remember that not all proxies insert comments in the via header indicating that they are proxies. The default value of the ignoreProxy property is `false`. If the checkViaHeader property is set to false, then the ignoreProxy property has no influence in establishing trust

-viaDepth
This property indicates a positive integer that specifies the number of source hosts in the via header to check for trust. By default, every host in the via header is checked, and if any host is not trusted, trust cannot be established. The viaDepth property is used only when some of the hosts in the via header have to be trusted. The setting indicates the number of hosts that are required to be trusted.

-ssoPwdExpiry
After trust is established for a request, the single sign-on user password is cached, eliminating the need to have the TAI re-authenticate the single sign-on user with Tivoli Access Manager for every request. You can modify the cache timeout period by setting the single sign-on password expiry property to the required time in seconds. If the password expiry property is set to 0, the cached password never expires. The default value for the password expiry property is 600.

-ignoreProxy
This property can be used to tell the TAI to ignore proxies as trusted hosts. If set to `true`, the

comments field of the hosts entry in the via header is checked to determine if a host is a proxy. Remember that not all proxies insert comments in the via header indicating that they are proxies. The default value of the ignoreProxy property is `false`. If the checkViaHeader property is set to false, then the ignoreProxy property has no influence in establishing trust.

-configURL
For the TAI to establish trust for a request, it requires that the SvrSslCfg task be run for the Java Virtual Machine on the Application Server and result in the creation of a properties file. If this properties file is not at the default URL, which is `file://java.home/PdPerm.properties`, the correct URL of the properties file must be set in the configuration URL property. If this property is not set, and the SvrSslCfg-generated properties file is not in the default location, the TAI initialization fails. The default value for the config URL property is `file://${WAS_INSTALL_ROOT}/java/jre/PdPerm.properties`.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configureTAMTAIProperties {-interactive}
```

- Using Jython:

```
AdminTask.configureTAMTAIProperties('-interactive')
```

unconfigureTAMTAIProperties

The unconfigureTAMTAIProperties command removes the custom properties from the security configuration for the embedded Tivoli Access Manager Trust Association Interceptor with classname TAMTrustAsociationInterceptorPlus.

Target object

None.

Required parameters

None.

Optional parameters

None.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask unconfigureTAMTAIProperties {-interactive}
```

- Using Jython:

```
AdminTask.unconfigureTAMTAIProperties('-interactive')
```

configureTAMTAIPdjrte

The configureTAMTAIPdjrte command performs the tasks necessary to fully configure the Tivoli Access Manager Runtime for Java. The specific tasks run are PDJrteCfg and SvrSslCfg.

Target object

None.

Required parameters

-policySvr
This property specifies the name of the Tivoli Access Manager policy server with which the application server communicates. The server is specified by fully qualified host name, the SSL port number, and the rank. The default SSL port number is 7135. For example: `myauth.mycompany.com:7135:1`.

-authSvrs
This property specifies the name of the Tivoli Access Manager authorization server with which the application server communicates. The server is specified by fully-qualified host name, the SSL port number, and the rank. The default SSL port number is 7136. For example: `myauth.mycompany.com:7136:1`. You can specify multiple servers if the entries are separated by a comma (,).

-adminPasswd
This property specifies the password of the Tivoli Access Manager administrator user that is associated with the -adminUid parameter. The password restrictions depend upon the password policy for your Tivoli Access Manager configuration.

Optional parameters

-adminUid
This property specifies the Tivoli Access Manager administrator name. If this option is not specified, `sec_master` is the default. A valid administrative ID is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the administrative ID.

For example, for U.S. English. the valid characters are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). The minimum and maximum lengths of the administrative ID, if there are limits, are imposed by the underlying registry.

-secDomain
This property specifies the Tivoli Access Manager domain name to which the administrator is authenticated. This domain must exist and an administrator ID and password must be valid for this domain. The application server is specified in this domain.

If this property is not specified, the default value is `Default`. The local domain value is retrieved from the configuration file.

A valid domain name is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the domain name.

For example, for U.S. English, the valid characters for domain names are the letters a-Z, the numbers 0-9, a period (.), an underscore (_), a plus sign (+), a hyphen (-), an at sign (@), an ampersand (&), and an asterisk (*). The minimum and maximum lengths of the domain name, if there are limits, are imposed by the underlying registry.

-defer
This property indicates whether this task should be run immediately or deferred until the startup of the WebSphere Application Server. The default value is `no`.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask configureTAMTAIPdjrte {-interactive}
```

- Using Jython:

```
AdminTask.configureTAMTAIPdjrte('-interactive')
```

unconfigureTAMTAIPdjrte

The unconfigureTAMTAIPdjrte command performs the tasks necessary to unconfigure the Tivoli Access Manager Runtime for Java. The specific tasks run are PDJrteCfg and SvrSslCfg.

Target object

None.

Required parameters

-adminPasswd
This property specifies the password of the Tivoli Access Manager administrator user that is associated with the -adminUid parameter. The password restrictions depend upon the password policy for your Tivoli Access Manager configuration.

Optional parameters

-adminUid
This property specifies the Tivoli Access Manager administrator name. If this option is not specified, `sec_master` is the default. A valid administrative ID is an alphanumeric, case-sensitive string. String values are expected to be characters that are part of the local code set. You cannot use a space in the administrative ID.

-force
This property indicates whether or not this task should stop when an error is encountered. The default value is `no`.

-defer
This property indicates whether this task should be run immediately or deferred until the startup of the WebSphere Application Server. The default value is `no`.

Examples

Interactive mode example usage:

- Using Jacl:

```
$AdminTask unconfigureTAMTAIPdjrte {-interactive}
```

- Using Jython:

```
AdminTask.unconfigureTAMTAIPdjrte('-interactive')
```

Disabling embedded Tivoli Access Manager client using the administrative console:

To unconfigure the JACC provider for Tivoli Access Manager, you can use the WebSphere Application Server administrative console.

Procedure

1. Click **Security > Global security > External authorization providers**.
2. Make sure that the default option, **Default authorization**, is checked, then click **OK**.
3. On the Global security panel, click **External authorization > External JACC provider**.
4. Under Additional properties, click **Tivoli Access Manager Properties**. The configuration screen for the JACC provider for Tivoli Access Manager is displayed.
5. Clear the **Enable embedded Tivoli Access Manager** option. If you want to ignore errors when unconfiguring, select the **Ignore errors during embedded Tivoli Access Manager disablement** option. Select this option only when the Tivoli Access Manager domain is in an irreparable state.
6. Click **OK**.
7. Optional: If you want security enabled without Tivoli Access Manager re-enable administrative security.

8. Restart all WebSphere Application Server instances for the changes to take effect.

Forcing the unconfiguration of the Tivoli Access Manager JACC provider:

If you find you cannot restart WebSphere Application Server after configuring the JACC provider for Tivoli Access Manager a utility is available to clear the security configuration and return WebSphere Application Server to an operable state.

About this task

The utility removes all of the PDLoginModuleWrapper entries as well as the Tivoli Access Manager authorization table from `security.xml` and `wsjaas.conf` files. This utility effectively removes the JACC provider for Tivoli Access Manager.

Procedure

1. Back up the `security.xml` and `wsjaas.conf` files.
2. Enter the following command as one continuous line.

```
java -classpath
"app_server_root/$WAS_HOME/plug-ins/com.tivoli.pd.amwas.core_6.1.0.jar"
com.tivoli.pd.as.jacc.cfg.CleanSecXML fully_qualified_path/security.xml
 fully_qualified_path/wsjaas.conf
```

Propagating security policies and roles for previously deployed applications:

Use this task to propagate security policies and roles to the external Java Authorization Contract for Containers (JACC) provider.

Before you begin

The external JACC provider must be configured before following these steps.

About this task

After switching to use the external JACC provider you can follow these steps to avoid having to redeploy your existing applications. Updating using these steps retrieves the security policy and roles from the deployed applications and propagates it to the external JACC provider removing the need for the applications to be redeployed.

Procedure

1. From the WebSphere Application Server administrative console, click **Security** > **Global security** > **External authorization providers**.
2. Select the appropriate security policy and role updating option.
   - Select **Don't update provider** to not propagate any security policies or roles
   - Select **Update with all applications** to propagate security policies and roles for all applications
   - Select **Update with application names listed** to propagate security policies and roles for the selected applications. If multiple applications should be updated, separate the application names with commas.
3. Click **Apply**.

Results

After completing this task your security policies and roles have been successfully propagated to the external JACC provider.

Authorizing access to administrative roles

You can assign users and groups to administrative roles to identify users who can perform WebSphere Application Server administrative functions.

Before you begin

Administrative roles enable you to control access to WebSphere Application Server administrative functions. Refer to the descriptions of these roles in “Administrative roles” on page 1623.

- Before you assign users to administrative roles, you must set up your user registry. For information on the supported registry types, see “Selecting a registry or repository” on page 1254.
- The following steps are needed to assign users to administrative roles.

About this task

You use the administrative console to assign users and groups to administrative roles and to identify users who can perform WebSphere Application Server administrative functions. In the administrative console,

Procedure

1. Click **Users and Groups**. Click either **Administrative User Roles** or **Administrative Group Roles**.
2. To add a user or a group, click **Add** on the Console users or Console groups panel.
3. To add a new administrator user, follow the instructions on the page to specify a user, and select the **Administrator** role. Once the user is added to the Mapped to role list, click **OK**. The specified user is mapped to the security role.
4. To add a new administrative group, follow the instructions on the page to specify either a group name or a Special subject, highlight the **Administrator** role, and click **OK**. The specified group or special subject is mapped to the security role.
5. To remove a user or group assignment, click **Remove** on the Console Users or the Console Groups panel. On the Console Users or the Console Groups panel, select the check box of the user or group to remove and click **OK**.
6. To manage the set of users or groups to display, click **Show filter function** on the User Roles or Group Roles panel. In the **Search term(s)** box, type a value, then click **Go**. For example, user* displays only users with the user prefix.
7. After the modifications are complete, click **Save** to save the mappings.
8. Restart the application server for changes to take effect.

Administrative user roles settings and CORBA naming service user settings

Use the Administrative User Roles page to give users specific authority to administer application servers through tools such as the administrative console or wsadmin scripting. The authority requirements are only effective when global security is enabled. Use the Common Object Request Broker Architecture (CORBA) naming service users settings page to manage CORBA naming service users settings.

To view the Console Users administrative console page, complete either of the following steps:

- Click **Security > Global security > Administrative User Roles**.
- Click **Users and Groups > Administrative User Roles**.

Note: If you are using local OS, the SIB administrative security panel's searches can use both the "?" and "*" search characters. However. if you switch to federated repositories, the searches will not work with the "?" character but will with the "*" character.

To view the CORBA naming service groups administrative console page, click **Environment > Naming > CORBA Naming Service Groups**.

Click **Refresh All** to automatically update the node agent and all of the nodes when a new user is created with the Administrator or Admin Security Manager role. When you click **Refresh All**, you do not need to manually restart the node agent under an existing Administrator before the new user is recognized with one of these roles. This button automatically invokes the AuthorizationManager refreshAll MBean method. To invoke this method manually, read about Fine-grained administrative security in heterogeneous and single-server environments.

User (Administrative user roles):

Specifies users.

The users that are entered must exist in the configured active user registry.

Data type: String

User (CORBA naming service users):

Specifies CORBA naming service users.

The users that are entered must exist in the configured active user registry.

Data type: String

Role (Administrative user roles):

Specifies user roles.

The following administrative roles provide different degrees of authority that are needed to perform certain application server administrative functions:

Administrator
The administrator role has operator permissions, configurator permissions, and the permission that is required to access sensitive data including server password, Lightweight Third Party Authentication (LTPA) password and keys, and so on.

Operator
The operator role has monitor permissions and can change the run-time state. For example, the operator can start or stop services.

Configurator
The configurator role has monitor permissions and can change the WebSphere Application Server configuration.

Deployer
The deployer role can complete both configuration actions and run-time operations on applications.

Monitor
The monitor role has the least permissions. This role primarily confines the user to viewing the application server configuration and current state.

adminsecuritymanager
The adminsecuritymanager role has privileges for managing users and groups from within the administrative console and determines who has access to modify users and groups using administrative role mapping. Only the adminsecuritymanager role can map users and groups to administrative roles, and by default, AdminId is granted to the adminsecuritymanager.

iscadmins
The iscadmins role has administrator privileges for managing users and groups from within the administrative console only.

Note: To manage users and groups, click **Users and Groups** in the console navigation tree. Click either **Manage Users** or **Manage Groups**.

Data type: String
Range: Administrator, Operator, Configurator, Deployer, Monitor, and iscadmins

Note: Other arbitrary administrative roles might also be visible in the administrative console collection table. Other contributors to the console might create these additional roles, which can be used for applications that are deployed to the console.

Role (CORBA naming service users):

Specifies naming service user roles.

A number of naming roles are defined to provide degrees of authority that are needed to perform certain application server naming service functions. The authorization policy is only enforced when global security is enabled. The following roles are valid: CosNamingRead, CosNamingWrite, CosNamingCreate, and CosNamingDelete.

The roles now have authority levels from low to high:

CosNamingRead
You can query the application server name space by using, for example, the Java Naming and Directory Interface (JNDI) lookup method. The EVERYONE special-subject is the default policy for this role.

CosNamingWrite
You can perform write operations such as JNDI bind, rebind, or unbind, plus CosNamingRead operations.

CosNamingCreate
You can create new objects in the name space through operations such as JNDI createSubcontext and CosNamingWrite operations.

CosNamingDelete
You can destroy objects in the name space, for example using the JNDI destroySubcontext method and CosNamingCreate operations.

Data type: String
Range: CosNamingRead, CosNamingWrite, CosNamingCreate and CosNamingDelete

Login status (Administrative user roles):

Specifies whether the user is active or inactive.

Administrative group roles and CORBA naming service groups

Use the Administrative Group Roles page to give groups specific authority to administer application servers through tools such as the administrative console or wsadmin scripting. The authority requirements are only effective when administrative security is enabled. Use the Common Object Request Broker Architecture (CORBA) naming service groups page to manage CORBA Naming Service groups settings.

To view the Console Groups administrative console page, complete either of the following steps:

- Click **Security > Global security > Administrative Group Roles**.
- Click **Users and Groups > Administrative Group Roles**.

To view the CORBA naming service groups administrative console page, click **Environment > Naming > CORBA Naming Service Groups**.

Click **Refresh All** to automatically update the node agent and all of the nodes when a new user is created with the Administrator or Admin Security Manager role. When you click **Refresh All**, you do not need to manually restart the node agent under an existing Administrator before the new user is recognized with one of these roles. This button automatically invokes the AuthorizationManager refreshAll MBean method. To invoke this method manually, read about Fine-grained administrative security in heterogeneous and single-server environments.

Group (CORBA naming service groups):

Identifies CORBA naming service groups.

In previous releases of WebSphere Application Server, there were two default groups: ALL AUTHENTICATED and EVERYONE. However, EVERYONE is now the only default group, and it provides CosNamingRead privileges only.

Data type: String
Range: EVERYONE

Role (CORBA naming service groups):

Identifies naming service group roles.

A number of naming roles are defined to provide the degrees of authority that are needed to perform certain application server naming service functions. The authorization policy is only enforced when global security is enabled.

Four name space security roles are available: CosNamingRead, CosNamingWrite, CosNamingCreate, and CosNamingDelete. The roles have authority levels from low to high:

Cos Naming Read
You can query the application server name space using, for example, the Java Naming and Directory Interface (JNDI) lookup method. The EVERYONE special-subject is the default policy for this role.

Cos Naming Write
You can perform write operations such as JNDI bind, rebind, or unbind, and CosNamingRead operations. The ALL_AUTHENTICATED special-subject is the default policy for this role.

Cos Naming Create
You can create new objects in the name space through operations such as JNDI createSubcontext and CosNamingWrite operations. The ALL_AUTHENTICATED special-subject is the default policy for this role.

Cos Naming Delete
You can destroy objects in the name space, for example using the JNDI destroySubcontext method and CosNamingCreate operations. The ALL_AUTHENTICATED special-subject is the default policy for this role.

Data type: String
Range: CosNamingRead, CosNamingWrite, CosNamingCreate, and CosNamingDelete

Group (Administrative group roles):

Specifies groups.

The ALL_AUTHENTICATED and the EVERYONE groups can have the following role privileges: Administrator, Configurator, Operator, and Monitor.

Data type:	String
Range:	ALL_AUTHENTICATED, EVERYONE

Role (Administrative group roles):

Specifies user roles.

The following administrative roles provide different degrees of authority needed to perform certain application server administrative functions:

Administrator
The administrator role has operator permissions, configurator permissions, and the permission that is required to access sensitive data, including server password, Lightweight Third Party Authentication (LTPA) password and keys, and so on.

Operator
The operator role has monitor permissions and can change the run-time state. For example, the operator can start or stop services.

Configurator
The configurator role has monitor permissions and can change the application server configuration.

Deployer
The deployer role can perform both configuration actions and runtime operations on applications.

Monitor
The monitor role has the least permissions. This role primarily confines the user to viewing the application server configuration and current state.

iscadmins
The iscadmins role has administrator privileges for managing users and groups from within the administrative console only.

Note: To manage users and groups, click **Users and Groups** in the console navigation tree. Click either **Manage Users** or **Manage Groups**.

Auditor
The auditor can view and modify the configuration settings for the security auditing subsystem. The auditor role includes the monitor role.

Data type:	String
Range:	Administrator, Operator, Configurator, Monitor, Deployer and iscadmins

Note: Other arbitrary administrative roles might also be visible in the administrative console collection table. Other contributors to the console might create these additional roles, which can be used for applications that are deployed to the console.

Assigning users to naming roles

Use this task to assign users to naming roles by using the administrative console.

About this task

The following steps are needed to assign users to naming roles. In the administrative console, click **Environment > Naming**, and click **CORBA Naming Service Users** or **CORBA Naming Service Groups**.

Procedure

1. Click **Add** on the CORBA Naming Service Users or the CORBA Naming Service Groups panel.

2. To add a new naming service user, follow the instructions on the page to specify a user, and select one or more roles. Once the user is added to the "Mapped to role" list, click **OK**. The specified user is mapped to one or more security roles.
3. To add a new naming service group, follow the instructions on the page to specify either a group name or a Special subject, highlight one or more roles, and click **OK**. The specified group or special subject are mapped to one or more the security roles
4. To remove a user or group assignment, go to the **CORBA Naming Service Users** or **CORBA Naming Service Groups** panel. Select the check box next to the user or group that you want to remove and click **Remove**.
5. To manage the set of users or groups to display, expand the **Filter** folder on the right panel, and modify the filter text box. For example, setting the filter to `user*` displays only users with the `user` prefix.
6. After modifications are complete, click **Save** to save the mappings. Restart the server for the changes to take effect.

Example

The default naming security policy is to grant all users read access to the CosNaming space and to grant any valid user the privilege to modify the contents of the CosNaming space. You can perform the previously mentioned steps to restrict user access to the CosNaming space. However, use caution when changing the naming security policy. Unless a Java Platform, Enterprise Edition (Java EE) application has clearly specified its naming space access requirements, changing the default policy can result in unexpected org.omg.CORBA.NO_PERMISSION exceptions at runtime.

Propagating administrative role changes to Tivoli Access Manager

These steps provide an example of how to migrate the `admin-authz.xml` file.

About this task

Additions and changes to console users and groups are not automatically added to the Tivoli Access Manager object space after the Java Authorization Contract for Containers (JACC) provider for Tivoli Access Manager is configured. Changes to console users and groups are saved in the `admin-authz.xml` file and this file must be migrated before any changes take effect. The JACC provider for Tivoli Access Manager includes the **migrateEAR** migration utility for incorporating console user and group changes into the Tivoli Access Manager object space.

Note: The **migrateEAR** utility is used to migrate the changes made to console users and groups after the JACC provider for Tivoli Access Manager is configured. The utility does not need to run for changes and additions to console users and groups made prior to the configuration of the JACC provider for Tivoli Access Manager because the changes made to the `admin-authz.xml` and `naming-authz.xml` files are automatically migrated at configuration time. Furthermore, the migration tool does not need to run before deploying standard Java Platform, Enterprise Edition (Java EE) applications; Java EE application policy deployment is also performed automatically.

For example, if you wanted to migrate the `admin-authz.xml` file, perform the following steps:

Procedure

1. Change to the *app_server_root*`/bin` directory where the `migrateEAR` utility is located.
2. Run the **migrateEAR** utility to migrate the data contained in the `admin-authz.xml` file. Use the parameter descriptions that are listed in “migrateEAR utility for Tivoli Access Manager” on page 1685.

   For example:

```
migrateEAR -profileName default
-j profile_root/config/cells/cell_name/xml_filename
-a sec_master
-p password
```

```
-w wsadmin
-d o=ibm,c=us
-c file:profile_root/etc/pd/PdPerm.properties
-z Roles
```

where *xml_filename* might be `admin-authz.xml` or `naming-authz.xml`.

- The `-j` parameter defaults to the file: *profile_root*`/config/cells/`*cell_name*`/admin-authz.html`
- The `-c` parameter defaults to the file: *profile_root*`/etc/pd/PdPerm.properties`. The output of the utility is logged in the `pdwas_migrate.log` file. The `pdwas_migrate.log` file is created in the *profile_root*`/logs` directory.
- The *-profile_name* parameter is optional and defaults to the default profile name.
- The `-z Roles` parameter is optional and when specified adds a subdirectory under the current directory structure in which to store the role mapping. For example,

```
/WebAppServer/deployedResouces/Roles
```

If `-z Roles` is not specified, the role mapping is stored in the current directory structure. For example,

```
/WebAppServer/deployedResouces
```

A status message is displayed when the migration completes. Output of the utility is logged to the `pdwas_migrate.log` file, which is created in the directory where the utility is run. Check the log file after each migration. If the log file displays errors, check the last recorded transaction, correct the source of the error, and rerun the migration utility. If the migration is unsuccessful, verify that you supplied the correct values for the `-c` and `-j` options.

3. WebSphere Application Server does not require a restart for the changes to take effect.

migrateEAR utility for Tivoli Access Manager

The **migrateEAR** utility migrates changes made to console users and groups in the `admin-authz.xml` and `naming-authz.xml` files into the Tivoli Access Manager object space.

Syntax

```
migrateEAR -profile_name default
-j fully_qualified_filename
-a Tivoli_Access_Manager_administrator_ID
-p Tivoli_Access_Manager_administrator_password
-w WebSphere_Application_Server_administrator_user_name
-d user_registry_domain_suffix
-c PdPerm.properties_file_location
[-z role_mapping_location]
```

Attention:

- The `-j` parameter defaults to the file: *profile_root*`/config/cells/`*cell_name*`/admin-authz.html`
- The `-c` parameter defaults to: `file:`*profile_root*`/etc/pd/PdPerm.properties`. The output of the utility is logged in the `pdwas_migrate.log` file. The `pdwas_migrate.log` file is created in the *profile_root*`/logs` directory.
- The *-profile_name* parameter is optional and defaults to the default profile name.

Parameters

-a*Tivoli_Access_Manager_administrator_ID*

The administrative user identifier. The administrative user must have the privileges required to create users, objects, and access control lists (ACLs). For example, `-a sec_master`.

This parameter is optional. When the parameter is not specified, you are prompted to supply it at run time.

-c *PdPerm.properties_file_location*

The Uniform Resource Indicator (URI) location of the `PdPerm.properties` file that is configured by the pdwascfg utility. When WebSphere Application Server is installed in the default location, the URI is:

file:*profile_root*/etc/pd/PdPerm.properties

-d *user_registry_domain_suffix*
The domain suffix for the user registry to use. For example, for Lightweight Directory Access Protocol (LDAP) user registries, this value is the domain suffix, such as: "o=ibm,c=us"

You can use the **pdadmin user show** command to display the distinguished name (DN) for a user.

-j *fully_qualified_pathname*
The fully qualified path and file name of the Java 2 Platform, Enterprise Edition application archive file ,admin-authz.xml or the roles definitions file naming-authz.xml that is used for a naming operation authorization. Optionally, this path can also be a directory of an expanded enterprise application. For example, when WebSphere Application Server is installed in the default location, the path to the data files to migrate includes:

profile_root/config/cells/*cell_name*

-p *Tivoli_Access_Manager_administrator_password*
The password for the Tivoli Access Manager administrative user. The administrative user must have the privileges that are required to create users, objects, and access control lists (ACLs). For example, you can specify the password for the -a sec_master administrative user as -p myPassword.

When this parameter is not specified, the user is prompted to supply the password for the administrative user name.

-w *WebSphere_Application_Server_administrator_user_name*
The user name that is configured in the WebSphere Application Server security user registry field as the administrator. This value matches the account that you created or imported in “Creating the security administrative user for Tivoli Access Manager” on page 1651. Access permission for this user is needed to create or update the Tivoli Access Manager protected object space.

When the WebSphere Application Server administrative user does not already exist in the protected object space, it is created or imported. In this case, a random password is generated for the user and the account is set to not valid. Change this password to a known value and set the account to valid.

A protected object and access control list (ACL) are created. The administrative user is added to the pdwas-admin group with the following ACL attributes:

T Traverse permission

i Invoke permission

WebAppServer
You can overwrite the action group name. The default name is WebAppServer. This action group name and the matching root object space can be overwritten when the migration utility is run with the **-r** option.

-z *role_mapping_location*
The location where the role mapping is to be stored when migrating administration applications. The default location is to place the role mapping in the current directory structure, such as:

/WebAppServer/deployedResouces

Specifying the -z option adds another directory level in which to store the role mapping. For example, if you specify -z Roles in the migrateEAR utility, the role mapping is stored in the directory structure as follows:

/WebAppServer/deployedResouces/Roles

Comments

This utility migrates security policy information from deployment descriptors or enterprise archive files to Tivoli Access Manager for WebSphere Application Server. The script calls com.tivoli.pdwas.migrate.Migrate the Java class.

Before you invoke the script, you must run the **setupCmdLine** script from the Qshell command line. You can find this file in the *profile_root*/bin directory, where *profile_root* is your installation path. In a default installation, *profile_root* is *app_server_rootBase*.

The script is dependent on finding the correct environment variables for the location of prerequisite software.

To enable a new user access to the administrative group in WebSphere Application Server, it is recommended that the user be added to the pdwas-admin group after JACC has been enabled. You can enter the administrative primary ID (adminID) in the group. This is required when the serverID is not the same as the adminID.

The following is an example of this command:

```
pdadmin> group modify pdwas-admin add adminID
```

Return codes

The utility can return the following exit status codes:

0 The command completed successfully.

1 The command failed.

Assigning users from a foreign realm to the admin-authz.xml

Operating with the administrative agent and job manager topology allows more situations where you might need to add an administrative user from a different registry into your administrative authorization table (admin-authz.xml). Each administrative user that needs to be added requires the "accessID" format of the user from the remote registry. When that user finally is active in the local cell, the authorization table will already have that accessID that is required. This task demonstrates how this assignment of users is performed.

Procedure

1. You need to determine the accessId for a user on the remote registry. To do this, you call the following wsadmin task and query based on a user filter. The following example illustrates a query from the registry realm "BIRKT60" with a userFilter of "localuser*". This query returns any user from this realm that begins with "localuser". The resulting accessId is the one you need to specify in the target administrative authorization table in the following step. Connect to the sending administrative process:

   ```
   wsadmin> $AdminTask listRegistryUsers {-securityRealmName BIRKT60 -displayAccessIds true -userFilter localuser*}
   {name BIRKT60\localuser@BIRKT60}
   {accessId user:BIRKT60/S-1-5-21-3033296400-14683092-2821094880-1007}
   ```

2. Add "localuser" to the target admin-authz.xml using the following wsadmin task. Connected to the receiving administrative process:

   ```
   wsadmin> $AdminTask mapUsersToAdminRole {-roleName administrator -userids {localuser }
   -accessids {user:BIRKT60/S-1-5-21-3033296400-14683092-2821094880-1007 }}
   ```

3. Save the changes.

Results

This task updates the admin-authz.xml in the receiving administrative process to allow a "cross-realm authorization" to succeed. The example illustrated here was for a LocalOS registry user. Performing the same task for an LDAP accessId produces results that look more like a realm and distinguished name (DN).

Note: If you change your realm you must repeat this process with the new realm name.

Fine-grained administrative security

In releases prior to WebSphere Application Server version 6.1, users granted administrative roles could administer all of the resources under the cell. WebSphere Application Server is now more fine-grained, meaning that access can be granted to each user per resource.

For example, users can be granted configurator access to a specific instance of a resource only (an application, an application server or a node). Users cannot access any other resources outside of the resources assigned to them. The administrative roles are now per resource rather than to the entire cell. However, there is a cell-wide authorization group for backward compatibility. Users assigned to administrative roles in the cell-wide authorization group can still access all of the resources within the cell.

Note: Nodes prior to WebSphere Application Server Version 6.1 in a mixed cell environment are filtered out of resource mapping.

To achieve this instance-based security or fine-grained security, resources that require the same privileges are placed in a group called the *administrative authorization group* or *authorization group*. Users can be granted access to the authorization group by assigning to them the required administrative role.

Fine-grained administrative security can also be used in single-server environments. Various applications in the single server can be grouped and placed in different authorization groups. Therefore, there are different authorization constraints for different applications. Note that the server itself cannot be part of any authorization group in a single-server environment.

You can assign users and groups to the adminsecuritymanager role on the cell level through wsadmin scripts and the administrative console. Using the adminsecuritymanager role, you can assign users and groups to the administrative user roles and administrative group roles.

When fine grained administrative security is used, users granted the adminsecuritymanager role can manage authorization groups. See “Administrative roles and naming service authorization” on page 1615 for detailed explanations of all administrative roles.

An administrator cannot assign users and groups to the administrative user roles and administrative group roles, including the adminsecuritymanager role. See “Administrative roles” on page 1623 for more details.

There are several administrative security commands that can be used to create authorization groups, map resources to authorization groups, and to assign users to administrative roles within the authorization groups. Following are some examples using wsadmin:

- **Create a new authorization group:**

```
$AdminTask createAuthorizationGroup {-authorizationGroupName authGroup1}
```

- **Deleting an authorization group:**

```
$AdminTask deleteAuthorizationGroup {-authorizationGroupName groupName}
```

- **Add resources to an authorization group:**

```
$AdminTask addResourceToAuthorizationGroup
{-authorizationGroupName groupName -resourceName Application=app1}
```

- **Remove resources from an authorization group:**

```
$AdminTask removeResourceFromAuthorizationGroup
{-authorizationGroupName groupName -resourceName Application=app1}
```

- **Add user IDs to roles in an authorization group:**

```
$AdminTask mapUsersToAdminRole {-authorizationGroupName groupName
-roleName administrator -userids user1}
```

- **Add group IDs to roles in an authorization group:**

```
$AdminTask mapGroupsToAdminRole {-authorizationGroupName groupName
-roleName administrator -groupids group1}
```

- **Remove user IDs from roles in an authorization group:**

```
AdminTask removeUsersFromAdminRole {-authorizationGroupName
groupName -roleName administrator -userids user1}
```

- **Remove group IDs from roles in an authorization group:**

```
$AdminTask removeGroupsFromAdminRole {-authorizationGroupName
groupName -roleName administrator -groupids group1}
```

Resources that can be added to an authorization group

You can add only resources of the following types to an authorization group:

- Cell
- Node
- ServerCluster
- Server
- Application
- NodeGroup

If a resource is not one of the types listed above, its parent resource will be used.

A resource can only belong to one authorization group. However, there is a containment relationship among resources. If a parent resource belongs to a different authorization group than that of its child resource, the child resource implicitly will belong to multiple authorization groups. You cannot add the same resource to more than one authorization group.

The following diagram shows the containment relationship among resources:

The privileges required for actions on resources depend on two factors:

- The authorization group of the administrative resource. If a user is granted access to an authorization group, all of the resources in that group will be included.
- The containment relationship of the resource. If a user is granted access to a parent resource, all of the children resources will be included.

Keystore management requires a user to have cell-level administrative privileges because they are created and managed at the cell level. Fine-grained security access to a specific resource does not allow management of the associated keystores.

Table 137. Privileges required to access various administrative resources. The privileges required to access various administrative resources are shown in the following table:

Resource	Action	Required roles
Server	Start, stop, runtime operations	Server-operator, node-operator, cell-operator
Server	New, delete	Node-configurator, cell-configurator
Server	Edit configuration	Server-configurator, node-configurator, cell-configurator
Server	View configuration, runtime status	Server-monitor, node-monitor, cell-monitor
Node	Restart, stop, sync	Node-operator, Cell-operator
Node	Add, delete	Cell-configurator
Node	Edit configuration	Node-configurator, cell-configurator
Node	View configuration, runtime status	Node-monitor, cell-monitor
Cluster	Start, stop, runtime operations	Cluster-operator, cell-operator
Cluster	New, delete	Cell-configurator
Cluster	Edit configuration	Cluster-configurator, cell-configurator

Table 137. Privileges required to access various administrative resources (continued). The privileges required to access various administrative resources are shown in the following table:

Resource	Action	Required roles
Cluster	View configuration, runtime status	Cluster-monitor, cell-monitor
Cluster member	Start, stop, runtime operations	Server-operator, cluster-operator, node-operator, cell-operator
Cluster member	New, delete	Node-configurator, cell-configurator
Cluster member	Edit configuration	Server-configurator, cluster-configurator, node-configurator, cell-configurator
Cluster member	View configuration, runtime status	Server-monitor, cluster-monitor, node-monitor, cell-monitor
Application	All operations	Refer to the section "Deployer roles" in “Administrative roles” on page 1623.
Node, cluster	Add, delete	Cell-configurator

The *server-operator* role is the operator role of the authorization group to which the server instance is part of. Similarly, the *node-operator* role is in the operator role of the authorization group to which the node instance is part of.

To use fine-grained administrative security in the administrative console, a user should be granted a monitor role at the cell level at minimum. However, to login using wsadmin, a user should be granted a monitor role for any authorization group.

Example: Using fine-grained security.

The following scenarios describe the use of fine-grained administrative security, particularly the new deployment role.

Deployment role scenario 1.

In the following scenario, there are four applications configured on server S1, as shown in the following table. Each application must be isolated so that the administrator of one application cannot modify another application. Assume that only user1 can manage application A1, user2 can manage applications A2 and A3, and only user3 can manage application A4.

Note: It is not recommended to have an application in one group and its target server in another group. However, that is not always possible. It is common to have many applications on one server. It is still sometimes necessary to isolate the administration of applications running on the same server.

One example is an Application Service Provider (ASP), where a single application server can have multiple vendor applications. In this instance the server administrator is responsible for installing all of the vendor applications. Once applications are installed, each vendor can manage their own application without interfering with other vendor's applications.

Table 138. Deployment role scenario 1 applications.

This table lists the Deployment role scenario 1 applications.

Application	Server	Node
A1	S1	N1
A2	S1	N1
A3	S1	N1
A4	S1	N1

We can configure authorization groups as shown in the diagram below:

In the diagram, application A1 is in authorization group G1, applications A2 and A3 are in authorization group G2, and application A4 is in authorization group G3.

A deployer role is assigned from authorization group G1 to user1, from authorization group G2 to user2, and from authorization group G3 to user3.

Consequently, user1 can perform all of the operations on application A1, user2 on applications A2 and A3, and user3 on application A4. Since all applications share the same server, we cannot put the same server on all authorization groups. Only a cell-level administrator can install an application. After the installation of an application is complete, the deployer of each application can modify their own. To start and stop the server, cell-level administrative authority is required. This type of scenario is useful in an ASP environment.

Deployment role scenario 2.

In the following scenario, a group of applications require the same administrative roles to one server. In this example, applications A1 and A2 are related applications, and can be administrated by one set of administrators. They are running on the same server (S1). Applications A3 and A4 require a different set of administrators, and are running on servers S2 and S3 respectively.

Table 139. Deployment role scenario 2 applications.

This table lists the Deployment role scenario 2 applications.

Application	Server	Node
A1	S1	N1
A2	S1	N1
A3	S2	N2
A4	S3	N3

Scenarios that can be applied directly in customer environments.

Each developer must be able to modify the configuration for their server, and they must be able to install their application onto that server. They also must be able to start and stop the server as well as the application on the server.

Developers also must be able to configure the server so that they can debug any problems they run into. They must have the ability to update or modify the application being developed. The administrative authorization group for this developer includes at least one server and any applications that the developer installs on that server.

In the following example, developers of authorization group G1 have a new application (A11). They can install and target that new application only on servers within authorization group G1. Also, they can place that new application in their authorization group (G1).

ASP environment scenario.

In this scenario, the customer is an ASP. They have their own customers to whom they provide application serving function. They want to enable their customers to administer and monitor their applications, but not to see or administer applications for different customers. In this example, however, the ASP has internal staff administrators whose job it is to maintain the servers.

This internal ASP staff administrator might need to move an application from one server to another to ensure that an application remains available. The internal ASP staff administrator should be able to stop and start the servers and to change their configuration.

In contrast, the ASP customer administrator should not be able to stop or start servers. However, the ASP customer administrator should be able to update their applications running on those servers. The administrative authorization group for the internal ASP administrator can be the whole cell or can include a subset of servers, nodes, clusters and applications. The administrative authorization group for the customer administrator only includes those applications that the customer has paid to have served by this ASP.

When updating the configuration repository, run the admin scripts from the deployment manager so that the fine grain admin security rules will be in effect when admin scripts are run from the deployment manager side.

The following diagram contains a scenario where two different customers have two different types of applications, and can manage their own applications. However, the servers and nodes on which the applications are running are isolated from their customers. The servers and nodes can only be maintained by the internal administrators. In addition, the customers cannot target their applications on a different server. This can only be performed by the internal administrator or internal deployers.

Regional organization scenario.

In this scenario, the customer is a large global company. The company's nodes and servers are organized so as to provide application serving for different regions (or alternatively, different lines of business). They want representatives from the different regional areas to be able to monitor and administer the nodes and servers associated with that region. However, they do not want the regional administer to be able to effect any node and server associated with a different region.

The administrative authorization group for each regional representative includes the nodes, servers, clusters and applications associated with that region.

For example, consider a company that provides multiple services, such as a financial institution that provides services like credit card accounts, brokerage accounts, banking accounts, or travel accounts. Each of these services can be separate applications, and the administrator for each of these applications must also be different. The following figure shows one way to configure such a system:

The following figure shows how the resources in such a system can be grouped to isolate administrators from each other:

Note that the nodes are not part of any authorization group. Therefore, a trade application administrator cannot stop a server on any of the nodes, and is prevented from stopping a travel application.

The same system can be configured in another way as shown below:

The following figure shows how the resources in such a system can be grouped to isolate administrators from each other:

New Administrative Authorization Group

Use this page to create a new administrative authorization group and to specify the associated administrative resources.

To view this administrative console page, click **Security** > **Administrative Authorization Groups** > **New**.

You must be logged into the administrative console with cell-level **AdminSecurityManager** authority, or the primary administrative ID can make these changes as well.

Name:

Use to identify the new administrative authorization group. The name should be descriptive of the group's role, or purpose, and should be unique in the cell structure. Using a non-unique name results in an error and a failure to create the new administrative authorization group. This is a required field.

Resources:

Select the resources from the Resource section to which you want the new administrative authorization group to control access.

Resources that are displayed in black text are available for selection.

Resources that are displayed in grey are already members of a different administrative authorization group. Therefore, these resources are not available for inclusion in the new administrative authorization group. When a resource is a member of a different authorization group, the name of the group displays next to the resource name. For example: server_1 (group_1)

The available filtering options are the following. Each option includes all the resources that are associated with that specific filtering option.

All scopes
The default view that displays the authorization group tree.

Clusters
All of the resources associated with the clusters.

Web Servers
All of the resources associated with the web servers.

Business-level applications
All of the resources associated with the business-level applications.

Servers
All of the resources associated with the servers.

Nodes
All of the resources associated with the nodes.

Applications
All of the resources associated with the applications.

Assets
All of the resources associated with the assets.

Node Groups
All of the resources associated with the node groups.

Assigned scopes
Displays all of the scopes explicitly assigned to the current authorization group.

Administrative Authorization Group collection
Use this page to create, delete or to edit an existing administrative authorization group.

To view this administrative console page, click **Security** > **Administrative Authorization Groups**.

You must be logged into the administrative console with cell-level **AdminSecurityManager** authority, or the primary administrative ID can make these changes as well.

Name:

The name field specifies the current name of the administrative authorization group. You can edit the name of the administrative authorization group during the creation process only. After the authorization group is created, you cannot modify the name. The specified name must be unique within the cell structure. Otherwise, a non-unique name results in an error.

New:

Select to create a new administrative authorization group.

Delete:

Select to remove an existing administrative authorization group.

Note: You must select an administrative authorization group before selecting **Delete**.

Creating a fine-grained administrative authorization group using the administrative console
You can create a fine-grained administrative authorization group by selecting administrative resources to be part of the authorization group. You can assign users or groups to this new administrative authorization group and also give them access to the administrative resources contained within.

Before you begin

You must be logged into the administrative console with cell-level **Admin Security Manager** authority, or the primary administrative ID can make these changes as well.

Procedure
1. Navigate to **Security** > **Administrative Authorization Groups** > **New**.
2. Type a name for the administrative authorization group into the **Name** field. This is a required field. The name must be unique within the cell structure. If the name is not unique then the new administrative authorization group is not created at the end of this procedure.
3. Select the resources from the Resource section to which you want the new administrative authorization group to control access.

   Resources that are displayed in black text are available for selection.

   Resources that are displayed in grey are already members of a different administrative authorization group. Therefore, these resources are not available for inclusion in the new administrative authorization group. When a resource is a member of a different authorization group, the name of the group displays next to the resource name. For example: server_1 (group_1)

Your filtering options include the following:

- Nodes. (All of the resources associated with the nodes.)
- Servers. (All of the resources associated with the servers.)
- Web servers. (All of the resources associated with the web servers.)
- Clusters. (All of the resources associated with the clusters.)
- Applications. (All of the resources associated with the applications.)
- Node groups. (All of the resources associated with the Node Groups.)
- All scopes. (The default view that displays the authorization group tree.)
- Assigned scopes. (Displays all of the scopes explicitly assigned to the current authorization group.)

4. Click **OK** or **Apply**.
5. If you want to associate a user role to this new administrative authorization group, do the following:
   a. Click **Administrative user roles** located under the Additional Properties section. The available user roles are the following:

      Administrator

      An individual or group that uses the administrator role has the operator and configurator privileges plus additional privileges that are granted solely to the administrator role. For example, an administrator can complete the following tasks:

      - Modify the server user ID and password.
      - Configure authentication and authorization mechanisms.
      - Enable or disable administrative security.
      - Enable or disable Java 2 security.
      - Change the Lightweight Third Party Authentication (LTPA) password and generate keys.
      - Create, update, or delete users in the federated repositories configuration.
      - Create, update, or delete groups in the federated repositories configuration.

      Note: An administrator cannot map users and groups to the administrator roles.
      .

      Configurator

      An individual or group that uses the configurator role has the monitor privilege plus the ability to change the WebSphere Application Server configuration. The configurator can perform all the day-to-day configuration tasks. For example, a configurator can complete the following tasks:

      - Create a resource.
      - Map an application server.
      - Install and uninstall an application.
      - Deploy an application.
      - Assign users and groups-to-role mapping for applications.
      - Set up Java 2 security permissions for applications.
      - Customize the Common Secure Interoperability Version 2 (CSIv2), Secure Authentication Service (SAS), and Secure Sockets Layer (SSL) configurations.

        Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

      Deployer

      Users granted this role can perform both configuration actions and runtime operations on applications..

Operator

An individual or group that uses the operator role has monitor privileges plus ability to change the runtime state. For example, an operator can complete the following tasks:

- Stop and start the server.
- Monitor the server status in the administrative console.

Monitor

An individual or group that uses the monitor role has the least amount of privileges. A monitor can complete the following tasks:

- View the WebSphere Application Server configuration.
- View the current state of the Application Server.

Admin Security Manager

Using the Admin Security Manager role, you can assign users and groups to the administrative user roles and administrative group roles. However, an administrator cannot assign users and groups to the administrative user roles and administrative group roles including the Admin Security Manager roles.

b. Click **Add...**. The New User page is displayed.

c. Select the appropriate role(s) from the **Role(s)** list box.

d. Select a user or users by entering text in the **Search string** field, and then click **Search.**. Click the arrow to add the available user or users to the **Mapped to role** field. You can select multiple users and roles by clicking **Select All**.

e. Click **OK**. You are returned to the Administrative User Roles page. The new users are displayed in the Administrative User Roles table along with their appropriate roles.

f. Repeat steps B through E for each new user to whom you want to map a role.

6. If you want to associate a group to this new user role, do the following:

   a. Click **Administrative group roles** located under the Additional Properties section.

   b. Click **Add...**. The New Group page is displayed.

   c. Select the appropriate role or roles from the **Role(s)** list box.

   d. Select a user or users by entering text in the **Search string** field, and then click **Search.**. Click the arrow to add the available user or users to the **Mapped to role** field. You can select multiple users and roles by clicking **Select All**.

   e. Select either the **Select from special subjects** or **Map Groups As Specified Below** option.

      If you select the **Select from special subjects** option, you can select the EVERYONE, ALL AUTHENTICATED, or ALL AUTHENTICATED IN TRUSTED REALMS values.

      A list of user groups and roles are displayed in the **Available** and **Mapped to role** fields. Select the user groups from the **Available** field and then select the roles from the **Mapped to role** field to which you want the group or groups associated. You can select multiple groups and roles.

   f. Click **OK**. You are returned to the Administrative Group Roles page. The new group is displayed in the Administrative Group Roles table along with the role of the new group.

   g. Repeat steps B through E for each new group to whom you want to map a role.

7. If you want to create another administrative authorization group, click **Apply**. The current administrative authorization group is created. Repeat steps 2 through 6 to create another administrative authorization group.

8. If you do not want to create another administrative authorization group, click **OK**.

Editing a fine-grained administrative authorization group using the administrative console

You can add or remove administrative resources to an administrative authorization group or edit an existing one.

Before you begin

You must be logged into the administrative console with the cell-level **AdminSecurityManager** authority or as the primary administrative user.

Procedure

1. Navigate to **Security** > **Administrative Authorization Groups**. The Administrative Authorization Groups page displays a table that lists all of the current administrative authorization groups available in the cell.
2. Click on the administrative authorization group in the table that you want to edit.
3. To add or remove resources from the administrative authorization group, select or clear them in the Resource section of the edit page. Resources displayed in black text are available for selection or clearing. Resources displayed in grey text are members of a different administrative authorization group and therefore cannot be edited for the current administrative authorization group.

   The available filtering options are the following. Each option includes all the resources that are associated with that specific filtering option.

   - All scopes. (The default view that displays the authorization group tree.)
   - Clusters. (All of the resources associated with the clusters.)
   - Web servers. (All of the resources associated with the Web servers.)
   - Business-level applications. (All of the resources associated with the business-level applications.)
   - Servers. (All of the resources associated with the servers.)
   - Nodes. (All of the resources associated with the nodes.)
   - Applications. (All of the resources associated with the applications.)
   - Assets. (All of the resources associated with the assets.)
   - Node groups. (All of the resources associated with the node groups.)
   - Assigned scopes. (Displays all of the scopes explicitly assigned to the current authorization group).

   Nodes prior to WebSphere Application Server Version 6.1 in a mixed cell environment are filtered out of resource mapping.
4. To remove a user or a group, do the following:
   a. To delete users, click **Administrative user roles** under the Additional Properties section. To delete groups, click **Administrative group roles** under the Additional Properties section. The appropriate edit page displays a table that lists all of the current users or groups and their associated roles, along with the user's login status.
   b. Click the check box beside the name of the current user or group and then click **Remove**. The current user or group is no longer associated with the role and the role is no longer listed in the table. It is now ready to have a new user or group assigned to it.
5. If you want to add or to reassign a user or group role to this administrative authorization group, do the following:
   a. To add a user, click **Administrative user roles** under the Additional Properties section. To add a group, click **Administrative group roles** located under the Additional Properties section. The appropriate edit page displays a table that lists all of the current users or groups and their associated roles. The available roles are:

      Administrator

      An individual or group that uses the administrator role has the operator and configurator privileges plus additional privileges that are granted solely to the administrator role. For example, an administrator can complete the following tasks:

      - Modify the server user ID and password.
      - Configure authentication and authorization mechanisms.
      - Enable or disable administrative security.

- Enable or disable Java 2 security.
- Change the Lightweight Third Party Authentication (LTPA) password and generate keys.
- Create, update, or delete users in the federated repositories configuration.
- Create, update, or delete groups in the federated repositories configuration.

Note: An administrator cannot map users and groups to the administrator roles.

Configurator
An individual or group that uses the configurator role has the monitor privilege plus the ability to change the WebSphere Application Server configuration. The configurator can perform all the day-to-day configuration tasks. For example, a configurator can complete the following tasks:

- Create a resource.
- Map an application server.
- Install and uninstall an application.
- Deploy an application.
- Assign users and groups-to-role mapping for applications.
- Set up Java 2 security permissions for applications.
- Customize the Common Secure Interoperability Version 2 (CSIv2), Security Authentication Service (SAS), and Secure Sockets Layer (SSL) configurations.

Important: SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.

Deployer
Users granted this role can perform both configuration actions and runtime operations on applications.

Operator
An individual or group that uses the operator role has monitor privileges plus ability to change the runtime state. For example, an operator can complete the following tasks:

- Stop and start the server.
- Monitor the server status in the administrative console.

.

Monitor
An individual or group that uses the monitor role has the least amount of privileges. A monitor can complete the following tasks:

- View the WebSphere Application Server configuration.
- View the current state of the Application Server.

Admin Security Manager
Using the Admin Security Manager role, you can assign users and groups to the administrative user roles and administrative group roles. However, an administrator cannot assign users and groups to the administrative user roles and administrative group roles including the Admin Security Manager role.

b. Click **Add...**.

c. To add a new user or group, follow the instructions on the page to specify either a user name, group name, or Special subject. Highlight the desired role(s), and click **OK**. The specified users, groups, or Special subject are mapped to the security roles.

Fine-grained administrative security in heterogeneous and single-server environments

Fine-grained administrative security can be used in heterogeneous or single-server environments with some restrictions.

Fine-grained administrative security in a heterogeneous environment

Fine-grained administrative security in a heterogeneous environment has the following restrictions:

- Only nodes that are running WebSphere Application Server Version 8.0 can be part of an administrative authorization group.
- Only servers that are running in a WebSphere Application Server Version 8.0 node can be part of an administrative authorization group.
- Only applications that are targeted on servers running on WebSphere Application Server Version 8.0 can be part of an administrative authorization group.
- If a cluster spans nodes of multiple releases, it cannot be part of an administrative authorization group.
- If a cluster spans nodes of multiple releases, none of its members can be part of an administrative authorization group.
- If an application is targeted on a cluster that spans multiple releases, that application cannot be part of an administrative authorization group.

Fine-grained administrative security in a single-server environment

You can also use fine-grained administrative security in a single-server environment. Various applications in the single server can be grouped and placed in different authorization groups. Therefore, different authorization constraints might exist for different applications.

Life cycle of fine-grained administrative resource

An administrative resource that was once part of an authorization group continues to be part of that authorization group until one of the following events occurs:

- The administrative resource is removed from the authorization group. In this instance, the administrative resource belongs to the cell-level authorization group.
- The administrative resource is removed from the configuration. In this instance, the administrative resource does not exist in the configuration, but still exists in the authorization group. Remove this administrative resource from the authorization group.

After the administrative resource is removed from the authorization group, the administrative authorizer runtime must be notified by using the AuthorizationManager refreshAll MBean method.

The **refreshAll** command must be invoked after AdminConfig.save() and sync nodes. For example:

JACL:

```
// get AuthorizationGroup Mbean
wsadmin> set agBean [$AdminControl queryNames
type=AuthorizationGroupManager,process=dmgr,*]

wsadmin> $AdminControl invoke &agBean refreshAll
```

JYTHON:

```
// get AuthorizationGroup Mbean
wsadmin> set agBean
AdminControl.queryNames('type=AuthorizationGroupManager,process=dmgr,*')

wsadmin> AdminControl.invoke(agBean, 'refreshAll')
```

The server restart is no longer needed.

Securing communications

WebSphere Application Server provides several methods to secure communication between a server and a client.

About this task

Note: WebSphere Application Server provides several methods for securing communication between a server and a client. New in this release are functions that ensure secure communication between a server and a client. These functions focus on certificate management, authentication, and ensuring trust among the application server, administrative agent, and job manager. The new functions include:

- Creating and using a certificate authority (CA) clients to enable a CA to request, query, and revoke certificates.
- Creating and using chained personal certificates to allow a certificate to be signed with a longer life span.
- Creating and revoking certificate authority (CA) certificates to ensure secure communication between the CA client and the CA server.

The following topics are covered in this section:

Procedure

- Secure communications using Secure Sockets Layer
- Creating an SSL configuration
- Creating a keystore configuration
- Creating a certificate authority (CA) client
- Deleting a certificate authority (CA) client
- Viewing or Modifying a certificate authority (CA) client
- Creating a keystore configuration for a preexisting keystore file
- Creating a self-signed certificate
- Creating a certificate authority request
- Extracting a signer certificate from a personal certificate
- Retrieving signers from a remote SSL port
- Adding a signer certificate to a keystore
- Adding a signer certificate to the default signers keystore
- Exchanging signer certificates in a keystore
- Configuring certificate expiration monitoring
- Key management for cryptographic uses
- Creating a key set configuration
- Creating a key set group configuration

Secure communications using Secure Sockets Layer (SSL)

The Secure Sockets Layer (SSL) protocol provides transport layer security including authenticity, data signing, and data encryption to ensure a secure connection between a client and server that uses WebSphere Application Server. The foundation technology for SSL is *public key cryptography*, which guarantees that when an entity encrypts data using its public key, only entities with the corresponding private key can decrypt that data.

WebSphere Application Server uses Java Secure Sockets Extension (JSSE) as the SSL implementation for secure connections. JSSE is part of the Java 2 Standard Edition (J2SE) specification and is included in

the IBM implementation of the Java Runtime Extension (JRE). JSSE handles the handshake negotiation and protection capabilities that are provided by SSL to ensure secure connectivity exists across most protocols. JSSE relies on X.509 certificate-based asymmetric key pairs for secure connection protection and some data encryption. Key pairs effectively encrypt session-based secret keys that encrypt larger blocks of data. The SSL implementation manages the X.509 certificates.

Managing X.509 certificates

Secure communications for WebSphere Application Server require digitally-signed X.509 certificates. The contents of an X.509 certificate, such as its distinguished name and expiration, are either signed by a certificate authority (CA), signed by a root certificate in **NodeDefaultRootStore** or **DmgrDefaultRootStore**, or are self-signed. When a trusted CA signs an X.509 certificate, WebSphere Application Server identifies the certificate and freely distributes it. A certificate must be signed by a CA because the certificate represents the identity of an entity to the general public. Server-side ports that accept connections from the general public must use CA-signed certificates. Most clients or browsers already have the signer certificate that can validate the X.509 certificate so signer exchange is not necessary for a successful connection.

You can trust the identity of a self-signed X.509 certificate only within a peer in a controlled environment, such as internal network communications, accepts the signer certificate. To complete a trusted handshake, you must first send a copy of the entity certificate to every peer that connects to the entity.

CA, chained, and self-signed X.509 certificates reside in Java *keystores*. JSSE provides a reference to the keystore in which a certificate resides. You can select from many types of keystores, including Java Cryptographic Extension (JCE)-based and hardware-based keystores. Typically, each JSSE configuration has two Java keystore references: a keystore and a *truststore*. The keystore reference represents a Java keystore object that holds personal certificates. The truststore reference represents a Java keystore object that holds signer certificates.

A personal certificate without a private key is an X.509 certificate that represents the entity that owns it during a handshake. Personal certificates contain both public and private keys. A signer certificate is an X.509 certificate that represents a peer entity or itself. Signer certificates contain just the public key and verify the signature of the identity that is received during a peer-to-peer handshake.

For more information, see “Extracting a signer certificate from a personal certificate” on page 1818

For more information about keystores, see “Keystore configurations for SSL” on page 1715.

Signer exchange

When you configure an SSL connection, you can exchange signers to establish trust in a *personal certificate* for a specific entity. Signer exchange enables you to extract the X.509 certificate from the peer keystore and add it into the truststore of another entity so that the two peer entities can connect. The signer certificate also can originate from a CA as a root signer certificate or a chained certificate's root signer certificate or an intermediate signer certificate. You can also extract a *signer certificate* directly from a self-signed certificate, which is the X.509 certificate with the public key.

Figure 1 illustrates a hypothetical keystore and truststore configuration. An SSL configuration determines which entities can connect to other entities, and the peer connections that are trusted by an SSL handshake. If you do not have the necessary signer certificate, the handshake fails because the peer cannot be trusted.

Figure 31. Signer exchange

In this example, the truststore for Entity A contains three signers. Entity A can connect to any peer as long as one of the three signers validates its personal certificate. For example, Entity A can connect to Entity B or Entity C because the signers can trust both signed personal certificates. The truststore for Entity-B contains one signer. Entity B is able to connect to Entity C only, and only when the peer endpoint is using certificate Entity-C Cert 1 as its identity. The ports that use the other personal certificate for Entity C are not trusted by Entity B. Entity C can connect to Entity A only.

In the example, the self-signed configuration seems to represent a one-to-one relationship between the signer and the certificate. However, when a CA signs a certificate, it typically signs many at a time. The advantage of using a single CA signer is that it can validate personal certificates that are generated by the CA for use by peers. However, if the signer is a public CA, you must be aware that the signed certificates might have been generated for another company other than your target entity. For your internal communications, private CAs and self-signed certificates are preferable to public CAs because they enable you to isolate the connections that you want to occur and prevent those that you do not want to occur.

SSL configurations

An SSL configuration comprises a set of configuration attributes that you can associate with an *endpoint* or set of endpoints in the WebSphere Application Server topology. The SSL configuration enables you to create an SSLContext object, which is the fundamental JSSE object that the server uses to obtain SSL socket factories. You can manage the following configuration attributes:

- An alias for the SSLContext object
- A handshake protocol version
- A keystore reference
- A truststore reference
- A key manager
- One or more trust managers
- A security level selection of a cipher suite grouping or a specific cipher suite list
- A certificate alias choice for client and server connections

To understand the specifics of each SSL configuration attribute, see “SSL configurations” on page 1706.

Selecting SSL configurations

In previous releases of WebSphere Application Server, you can reference an SSL configuration only by selecting the SSL configuration alias directly. Each secure endpoint was denoted by an alias attribute that references a valid SSL configuration within a repertoire of SSL configurations. When you made a single configuration change, you had to re-configure many alias references across the various processes. Although the current release still supports direct selection, this approach is no longer recommended.

The current release provides improved capabilities for managing SSL configurations and more flexibility when you select SSL configurations. In this release, you can select from the following approaches:

Programmatic selection

You can set an SSL configuration on the running thread prior to an outbound connection. WebSphere Application Server ensures that most system protocols, including Internet Inter-ORB Protocol (IIOP), Java Message Service (JMS), Hyper Text Transfer Protocol (HTTP), and Lightweight Directory Access Protocol (LDAP), accept the configuration. See “Programmatically specifying an outbound SSL configuration using JSSEHelper API” on page 1772

Dynamic selection

You can associate an SSL configuration dynamically with a specific target host, port, or outbound protocol by using a predefined selection criteria. When it establishes the connection, WebSphere Application Server checks to see if the target host and port match a predefined criteria that includes the domain portion of the host. Additionally, you can predefine the protocol for a specific

outbound SSL configuration and certificate alias selection. See “Dynamic outbound selection of Secure Sockets Layer configurations” on page 1717 for more information.

Dynamic outbound selection of Secure Sockets Layer configurations is based on connection information being available for the server so that the server can match up the outbound protocol, host, or port when the creation of the client socket takes place in com.ibm.websphere.ssl.protocol.SSLSocketFactory. For WebSphere admin connectors like SOAP and Remote Method Invocation (RMI), connection information is placed on the thread and is available for dynamic outbound selection to take place. The dynamic outbound selection process replies on connection information being setup when SSL properties are retrieved similar to what is described in “Programmatically specifying an outbound SSL configuration using JSSEHelper API” on page 1772.

When the outbound connection is being made from customer written applications, parts of the connection information may not be available. Some of these applications make API calls to a protocol to make the connection. The API ultimately then calls one of the createSocket() methods in com.ibm.websphere.ssl.protocol.SSLSocketFactory to complete the process. gotcha: All of the connection information for dynamic outbound selection might not be available, and you may have to adjust the dynamic outbound selection connection filter and fill in an asterisk (*) for the missing part of the connection information. As an example, the openConnection() call on a URL object ultimately calls `createSocket(java.net.Socket socket, String host, int port, boolean autoClose)`. The connection information can be built with the host and port provided, but there is no protocol provided. In this case, a wild card, asterisk (*), should be used for the protocol part of the dynamic selection connection information.

Most of the createSocket() methods take a host or IP address and a port as parameters. The dynamic outbound selection connection filter can be built with the host and port. The default method, createSocket(), without any parameters does not contain any information to build the outbound selection connection filter unless the socket factory was instantiated with connection information, If no connection information is available, then you should consider using one of the other methods of selecting a SSL configuration describes in this topic, "Programmatic selection" can be good choice.

Direct selection

You can select an SSL configuration by using a specific alias, as in past releases. This method of selection is maintained for backwards compatibility because many applications and processes rely on alias references.

Scope selection

You can associate an SSL configuration and its certificate alias, which is located in the keystore associated with that SSL configuration, with a WebSphere Application Server management scope. This approach is recommended to manage SSL configurations centrally. You can manage endpoints more efficiently because they are located in one topology view of the cell. The inheritance relationship between scopes reduces the number of SSL configuration assignments that you must set.

Each time you associate an SSL configuration with a cell scope, the node scope within the cell automatically inherits the configuration properties. However, when you assign an SSL configuration to a node, the node configuration overrides the configuration that the node inherits from the cell. Similarly, all of the application servers for a node automatically inherit the SSL configuration for that node unless you override these assignments. Unless you override a specific configuration, the topology relies on the rules of inheritance from the cell level down to the endpoint level for each application server.

Note: If your applications are relying on SSL configurations that were set as individual settings for each SSL configuration in the topology, but your application servers have inherited the SSL configuration as deployed from the cell level down to the endpoint level, then there is the

possibility of communication errors occurring among servers (for example, handshake errors). You need to ensure that your applications are operating consistent with the central management of SSL configurations.

The topology view displays an inbound tree and outbound tree. You can make different SSL configuration selections for each side of the SSL connection based on what that server connects to as an outbound connection and what the server connects to as an inbound connection. See "Central management of SSL configurations" on page 1718 for more information.

The runtime uses an order of precedence for determining which SSL configuration to choose because you have many ways to select SSL configurations. Consider the following order of precedence when you select a configuration approach:

1. Programmatic selection. If an application sets an SSL configuration on the running thread using the com.ibm.websphere.ssl.JSSEHelper application programming interface (API), the SSL configuration is guaranteed the highest precedence.
2. Dynamic selection criteria for outbound host and port or protocol.
3. Direct selection.
4. Scope selection. Scope inheritance guarantees that the endpoint that you select is associated with an SSL configuration and is inherited by every scope beneath it that does not override this selection.

Default chained certificate configuration

By default, WebSphere Application Server creates a unique chained certificate for each node. The chained certificate is signed with a root, a self-signed certificate stored in the **DmgrDefaultRootStore** or **NodeDefaultRootStore**. WebSphere Application Server no longer relies on a self-signed certificate or the default or dummy certificate that is shipped with the product. The `key.p12` default keystore and the trust.p12 truststore are stored in the configuration repository within the node directory. The default root certificate is stored in the root-key.p12 in the configuration repository under the node directory.

All of the nodes put their signer certificates from the default root certificate in this common truststore (`trust.p12`). Additionally, after you federate a node, the default SSL configuration is automatically modified to point to the common truststore, which is located in the cell directory. The node can now communicate with all other servers in the cell.

All default SSL configurations contain a keystore with the name suffix DefaultKeyStore, a truststore with the name suffix DefaultTrustStore and a rootstore with the name suffix DefaultRootStore. These default suffixes instruct the WebSphere Application Server runtime to add the root signer of the personal certificate to the common truststore. If a truststore name does not end with DefaultKeyStore, the keystores root signer certificates are not added to the common truststore when you federate the server. You can change the default SSL configuration, but you must ensure that the correct trust is established for administrative connections, among others.

For more information, see "Default chained certificate configuration in SSL" on page 1724.

Certificate expiration monitoring

Certificate monitoring ensures that the default chained certificate for each node is not allowed to expire. The certificate expiration monitoring function issues a warning before certificates and signers are set to expire. Those certificates and signers that are located in keystores managed by the WebSphere Application Server configuration can be monitored. You can configure the expiration monitor to automatically replace a certificate. A chained certificate will be recreated based on the same data used for the initial creation and sign it with the same root certificate that signed the original certificate. A self-signed certificate or chained certificate is also recreated based upon the same data that is used for the initial creation.

The monitor also can automatically replace old signers with the signers from the new chained or self-signed certificates in keystores that are managed by WebSphere Application Server. The existing signer exchanges that occurred by the runtime during federation and by administration are preserved. For more information, see “Certificate expiration monitoring in SSL” on page 1732.

The expiration monitor is configured to replace chained personal certificates that are signed by a root certificate in **DmgrDefaultRootStore** or **NodeDefaultRootStore**. The certificate is renewed using the same root certificate that was used to sign the original certificate.

The monitor also can automatically replace old signers with the signers from the new self-signed certificates in keystores that are managed by WebSphere Application Server. The existing signer exchanges that occurred by the runtime during federation and by administration are preserved. For more information, see “Certificate expiration monitoring in SSL” on page 1732.

WebSphere Application Server clients: signer-exchange requirements

A new chained certificate is generated for each node during its initial startup. To ensure trust, clients must be given the root signers to establish a connection. The introduction of chained certificates in the current release makes this process simpler. Rather than exchanging the signer of a short lived self-signed certificate, you can exchange the long lived root signer which will allow for preserved trust across personal certificate renewals. In addition, you can gain access to the signer certificates of various nodes to which the client must connect with any one of the following options (for more information, see “Secure installation for client signer retrieval in SSL” on page 1727):

- A signer exchange prompt enables you to import signer certificates that are not yet present in the truststores during a connection to a server. By default, this prompt is enabled for administrative connections and can be enabled for any client SSL configuration. When this prompt is enabled, any connection that is made to a server where the signer is not already present offers the signer of the server along with the certificate information and a Secure Hash Algorithm (SHA) digest of the certificate for verification. The user is given a choice whether to accept these credentials. If the credentials are accepted, the signer is added to the truststore of the client until the signer is explicitly removed. The signer exchange prompt does not occur again when connecting to the same server unless the personal certificate changes.

  Attention: It is unsafe to trust a signer exchange prompt without verifying the SHA digest. An unverified prompt can originate from a browser when a certificate is not trusted.
- You can run a retrieveSigners administrative script from a client prior to making connections to servers. To download signers, no administrative authority is required. To upload signers, you must have Administrator role authority. The script downloads all of the signers from a specified server truststore into the specified client truststore and can be called to download only a specific alias from a truststore. You can also call the script to upload signers to server truststores. When you select the CellDefaultTrustStore truststore as the specified server truststore and common truststore for a cell, all of the signers for that cell are downloaded to the specified client truststore, which is typically ClientDefaultTrustStore. For more information, see “retrieveSigners command” on page 1730.
- You can physically distribute to clients the trust.p12 common truststore that is located in the cell directory of the configuration repository. When doing this distribution, however, you must ensure that the correct password has been specified in the `ssl.client.props` client SSL configuration file. The default password for this truststore is `WebAS`. Change the default password prior to distribution. Physical distribution is not as effective as the previous options. When changes are made to the personal certificates on the server, automated exchange can fail.

Dynamic SSL configuration changes

The SSL runtime for WebSphere Application Server maintains listeners for most SSL connections. A change to the SSL configuration causes the inbound connection listeners to create a new SSLContext object. Existing connections continue to use the current SSLContext object. Outbound connections automatically use the new configuration properties when they are attempted.

Make dynamic changes to the SSL configuration during off-peak hours to reduce the possibility of timing-related problems and to prevent the possibility of the server starting again. If you enable the runtime to accept dynamic changes, then change the SSL configuration and save the `security.xml` file. Your changes take effect when the new `security.xml` file reaches each node.

Note: If configuration changes cause SSL handshake failures, administrative connectivity failures also can occur, which can lead to outages. In this case, you must re-configure the SSL connections then perform manual node synchronization to correct the problem. You must carefully complete any dynamic changes. It is highly recommended that you perform changes to SSL configurations on a test environment prior to making the same changes to a production system. For more information, see “Dynamic configuration updates in SSL” on page 1735.

Built-in certificate management

Certificate management that is comparable to iKeyMan functionality is now integrated into the keystore management panels of the administrative console. Use built-in certificate management to manage personal certificates, certificate requests, and signer certificates that are located in keystores. Additionally, you can remotely manage keystores. For example, you can manage a file-based keystore that is located outside the configuration repository on any node from the deployment manager. You also can remotely manage hardware cryptographic keystores from the deployment manager.

With built-in certificate management, you can replace a chained or self-signed certificate along with all of the signer certificates scattered across many truststores and retrieve a signer from a remote port by connecting to the remote SSL host and port and intercepting the signer during the handshake. The certificate is first validated according to the certificate SHA digest, then the administrator must accept the validated certificate before it can be placed into a truststore.

When you make a certificate request, you can send it to a certificate authority (CA). When the certificate is returned, you can accept it within the administrative console. For more information, see “Certificate management in SSL” on page 1736.

Tip: Although iKeyMan functionality still ships with WebSphere Application Server, configure keystores from the administrative console using the built-in certificate management functionality. iKeyMan is still an option when it is not convenient to use the administrative console. For more information, see “Certificate management using iKeyman prior to SSL” on page 1735.

AdminTask configuration management

The SSL configuration management panels in the administrative console rely primarily on administrative tasks, which are maintained and enhanced to support the administrative console function. You can use wsadmin commands from a Java console prompt to automate the management of keystores, SSL configurations, certificates, and so on.

SSL configurations

Secure Sockets Layer (SSL) configurations contain attributes that enable you to control the behavior of both the client and the server SSL endpoints. You can assign SSL configurations to have specific management scopes. The scope that an SSL configuration inherits depends upon whether you create it using a cell, node, server, or endpoint link in the configuration topology.

When you create an SSL configuration, you can set the following SSL connection attributes:

- Keystore
- Default client certificate for outbound connections
- Default server certificate for inbound connections
- Truststore
- Key manager for selecting a certificate

- Trust manager or managers for establishing trust during the handshake
- Handshaking protocol
- Ciphers for negotiating the handshake
- Client authentication support and requirements

You can manage an SSL configuration using any of the following methods:
- Central management selection
- Direct reference selection
- Dynamic outbound connection selection
- Programmatic selection

Using the administrative console, you can manage all of the SSL configurations for WebSphere Application Server. From the administrative console, click **Security > SSL certificates and key management > Manage endpoint security configurations > Inbound | Outbound >** *SSL_configuration*. You can view an SSL configuration at the level it was created and in the inherited scope below that point in the topology. If you want the entire cell to view an SSL configuration, you must create the configuration at the cell level in the topology.

SSL configuration in the security.xml file

The attributes defining an SSL configuration repertoire entry for a specific management scope are stored in the `security.xml` file. The scope determines the point at which other levels in the cell topology can see the configuration, as shown in the following example:

```
<repertoire xmi:id="SSLConfig_1" alias="NodeDefaultSSLSettings"
managementScope="ManagementScope_1" type="JSSE">
<setting xmi:id="SecureSocketLayer_1" clientAuthentication="false"
clientAuthenticationSupported="false" securityLevel="HIGH" enabledCiphers=""
jsseProvider="IBMJSSE2" sslProtocol="SSL_TLS" keyStore="KeyStore_1"
trustStore="KeyStore_2" trustManager="TrustManager_1" keyManager="KeyManager_1"
clientKeyAlias="default" serverKeyAlias="default"/>
</repertoire>
```

The SSL configuration attributes from the previous code sample are described in Table 1.

Table 140. security.xml Attributes. This table lists the security.xml Attributes.

security.xml attribute	Description	Default	Associated SSL property
xmi:id	The xml:id attribute represents the unique identifier for this XML entry and determines how the SSL configuration is linked to other XML objects, such as SSLConfigGroup. This system-defined value must be unique.	The administrative configuration service defines the default value.	None. This value is used only for XML associations.
alias	The alias attribute defines the name of the SSL configuration. Direct selection uses the alias attribute and the node is not prefixed to the alias. Rather, the management scope takes care of ensuring that the name is unique within the scope.		com.ibm.ssl.alias
managementScope	The managementScope attribute defines the management scope for the SSL configuration and determines the visibility of the SSL configuration at runtime.		The managementScope attribute is not mapped to an SSL property. However, it confirms whether or not the SSL configuration is associated with a process.

Table 140. security.xml Attributes (continued). This table lists the security.xml Attributes.

security.xml attribute	Description	Default	Associated SSL property
type	The type attribute defines the Java Secure Socket Extension (JSSE) or System Secure Sockets Layer (SSSL) configuration option. JSSE is the SSL configuration type for most secure communications within WebSphere Application Server.	The default is JSSE.	com.ibm.ssl.sslType
clientAuthentication	The clientAuthentication attribute determines whether SSL client authentication is required.	The default is false.	com.ibm.ssl.clientAuthentication
clientAuthenticationSupported	The clientAuthenticationSupported attribute determines whether SSL client authentication is supported. The client does not have to supply a client certificate if it does not have a client certificate. **Attention:** When you set the clientAuthentication attribute to `true`, you override the value that is set for the clientAuthenticationSupported attribute.	The default is false.	com.ibm.ssl.client.AuthenticationSupported
securityLevel	The securityLevel attribute determines the cipher suite group. Valid values include STRONG (128-bit ciphers), MEDIUM (40-bit ciphers), WEAK (for all ciphers without encryption), and CUSTOM (if the cipher suite group is customized. When you set the enabledCiphers attribute with a specific list of ciphers, the system ignores this attribute.	The default is STRONG.	com.ibm.ssl.securityLevel
enabledCiphers	You can set the enabledCiphers attribute to specify a unique list of cipher suites. Separate each cipher suite in the list with a space.	The default is the securityLevel attribute for cipher suite selection.	com.ibm.ssl.enabledCipherSuites
jsseProvider	The jsseProvider attribute defines a specific JSSE provider.	The default is IBMJSSE2.	com.ibm.ssl.contextProvider
sslProtocol	The sslProtocol attribute defines the SSL handshake protocol. Valid options include: SSLv2 (client-side only), SSLv3, SSL, SSL_TLS, TLSv1, and TLS values. The SSL option includes SSLv2 and SSLv3 values. The TLS option includes the TLSv1 value. SSL_TLS, which is the most interoperable protocol, includes all these values and defaults to a Transport Layer Security (TLS) handshake.	The default is SSL_TLS.	com.ibm.ssl.protocol

Table 140. security.xml Attributes (continued). This table lists the security.xml Attributes.

security.xml attribute	Description	Default	Associated SSL property
keyStore	The keyStore attribute defines the keystore and attributes of the keyStore instance that the SSL configuration uses for key selection.		For more information, see Keystore configurations.
trustStore	The trustStore attribute defines the key store that the SSL configuration uses for certificate signing verification.		A trustStore is a logical JSSE term. It signifies a key store that contains signer certificates. Signer certificates validate certificates that are sent to WebSphere Application Server during an SSL handshake.
keyManager	The keyManager attribute defines the key manager that WebSphere Application Server uses to select keys from a key store. A JSSE key manager controls the javax.net.ssl.X509KeyManager interface. A custom key manager controls the javax.net.ssl.X509KeyManager and the com.ibm.wsspi.ssl.KeyManagerExtendedInfo interfaces. The com.ibm.wsspi.ssl.KeyManagerExtendedInfo interface provides more information from WebSphere Application Server.	The default is IbmX509.	com.ibm.ssl.keyManager defines a well-known key manager and accepts the algorithm and algorithm|provider formats, for example IbmX509 and IbmX509|IBMJSSE2. com.ibm.ssl.customKeyManager defines a custom key manager and takes precedence over the other keyManager properties. This class must implement javax.net.ssl.X509KeyManager and can implement com.ibm.wsspi.ssl.KeyManagerExtendedInfo. For more information, see "Key manager control of X.509 certificate identities" on page 1711
trustManager	The trustManager determines which trust manager or list of trust managers to use for determining whether to trust the peer side of the connection. A JSSE trust manager implements the javax.net.ssl.X509TrustManager interface. A custom trust manager might also implement com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface to get more information from the WebSphere Application Server environment.	The default is IbmPKIX, which can be configured for certificate revocation list (CRL) verification when the certificate contains a CRL distribution point. The other option is IbmX509.	com.ibm.ssl.trustManager defines a well-known trust manager, which is required for most handshake situations. com.ibm.ssl.trustManager performs certificate expiration checking and signature validation. You can define com.ibm.ssl.customTrustManagers with additional custom trust managers that are called during an SSL handshake. Separate additional trust managers with the vertical bar (|) character. For more information, see "Trust manager control of X.509 certificate trust decisions"

Client SSL configurations are managed using the `ssl.client.props` properties file. The `ssl.client.props` file is located in the ${USER_INSTALL_ROOT}/properties directory for each profile. For more information about configuring this file, see the "ssl.client.props client configuration file" on page 1783. Specifying any javax.net.ssl system properties will override the corresponding property in the `ssl.client.props` file.

Trust manager control of X.509 certificate trust decisions:

The role of the trust manager is to validate the Secure Sockets Layer (SSL) certificate that is sent by the peer, which includes verifying the signature and checking the expiration date of the certificate. A Java Secure Socket Extension (JSSE) trust manager determines if the remote peer can be trusted during an SSL handshake.

WebSphere Application Server has the ability to call multiple trust managers during an SSL connection. The default trust manager does the standard certificate validation; custom trust manager plug-ins run customized validation such as host name verification. For more information, see "Example: Developing a custom trust manager for custom SSL trust decisions" on page 1765

When a trust manager is configured in a server-side SSL configuration, the server calls the isClientTrusted method. When a trust manager is configured in a client-side SSL configuration, the client calls the isServerTrusted method. The peer certificate chain is passed to these methods. If the trust manager chooses not to trust the peer information, it might produce an exception to force a handshake failure.

Optionally, WebSphere Application Server provides the com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface so that additional information can be passed to the trust manager. For more information, see the com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface.

Default IbmX509 trust manager

The default IbmX509 trust manager, which is used in the following code sample, establishes trust by performing standard certificate validation.

```
<trustManagers xmi:id="TrustManager_1132357815717" name="IbmX509" provider="IBMJSSE2"
algorithm="IbmX509" managementScope="ManagementScope_1132357815717"/>
```

The trust manager provides a signer certificate to verify the peer certificate that is sent during the handshake. The signers who are added to the truststore for the SSL configuration must be trustworthy. If you do not trust the signers or do not want to allow others to connect to your servers, consider removing default root certificates from certificate authorities (CA). You might also remove any certificates if you cannot verify their origination.

Default IbmPKIX trust manager

You can use the default IbmPKIX trust manager to replace the IbmX509 trust manager, which is shown in the following code sample:

```
<trustManagers xmi:id="TrustManager_1132357815719" name="IbmPKIX" provider="IBMJSSE2"
algorithm="IbmPKIX" trustManagerClass="" managementScope="ManagementScope_1132357815717">
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1132357815717"
name="com.ibm.security.enableCRLDP" value="true" type="boolean"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1132357815718"
name="com.ibm.jsse2.checkRevocation" value="true" type="boolean"/>
  </trustManagers>

<trustManagers xmi:id="TrustManager_managementNode_2" name="IbmPKIX" provider=
"IBMJSSE2" algorithm="IbmPKIX" trustManagerClass=""
managementScope="ManagementScope_managementNode_1">
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1" name="com.ibm.se
curity.enableCRLDP" value="false" type="boolean" displayNameKey="" nlsRangeKey="
" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_2" name="com.ibm.js
se2.checkRevocation" value="false" type="boolean" displayNameKey="" nlsRangeKey=
"" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_3" name="ocsp.enabl
e" value="false" type="String" displayNameKey="" nlsRangeKey="" hoverHelpKey=""
range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_4" name="ocsp.respo
nderURL" value="http://ocsp.example.net:80" type="String" displayNameKey=""
nlsRangeKey="" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_5" name="ocsp.respo
nderCertSubjectName" value="" type="String" displayNameKey="" nlsRangeKey="" hov
erHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_6" name="ocsp.respo
nderCertIssuerName" value="" type="String" displayNameKey="" nlsRangeKey="" hove
rHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_7" name="ocsp.respo
nderCertSerialNumber" value="" type="String" displayNameKey="" nlsRangeKey="" ho
verHelpKey="" range="" inclusive="false" firstClass="false"/>
</trustManagers>
```

See “Example: Enabling certificate revocation checking with the default IbmPKIX trust manager” on page 1712 for additional information in using the default IbmPKIX trust manager.

In addition to its role of standard certificate verification, the IbmPKIX trust manager checks for OCSP properties and for certificates that contain certificate revocation list (CRL) distribution points. This process is known as extended CRL checking. When you select a trust manager, its associated properties are automatically set as Java System properties so that the IBMCertPath and IBMJSSE2 providers are aware that CRL checking is enabled.

Custom trust manager

You can define a custom trust manager to perform additional trust checking, which is based upon the needs of the environment. For example, in one environment, you might enable connections from the same Transmission Control Protocol (TCP) subnet only. The com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface provides extended information about the connection that is not provided by the standard Java Secure Sockets Extension (JSSE) javax.net.ssl.X509TrustManager interface. The configured trustManagerClass attribute determines which class is instantiated by the runtime, as shown in the following code sample:

```
<trustManagers xmi:id="TrustManager_1132357815718" name="CustomTrustManager"
trustManagerClass="com.ibm.ws.ssl.core.CustomTrustManager"
managementScope="ManagementScope_1132357815717"/>
```

The trustManagerClass attribute must implement the javax.net.ssl.X509TrustManager interface and, optionally, can implement the com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface.

Disabling the default trust manager

In some cases, you might not want to perform the standard certificate verification that is provided by the IbmX509 and IbmPKIX default trust managers. For example, you might be working with an internal automated test infrastructure that is not concerned with SSL client or server authentication, integrity, or confidentiality. The following sample code shows a basic custom trust manager such as com.ibm.ws.ssl.core.CustomTrustManager whose property is set to `true`.

```
com.ibm.ssl.skipDefaultTrustManagerWhenCustomDefined=true
```

You can set this property in the global properties at the top of the `ssl.client.props` file for clients or in the `security.xml` custom properties file for servers. You must configure a custom trust manager when you disable the default trust manager to prevent the server from calling the default trust manager even though it is configured. Disabling the default trust manager is not a common practice. Be sure to test the system with the disabled default trust manager in a test environment first. For more information on setting up a custom trust manager, see “Creating a custom trust manager configuration for SSL” on page 1761

Key manager control of X.509 certificate identities:

The role of a Java Secure Socket Extension (JSSE) key manager is to retrieve the certificate that is used to identify the client or server during a Secure Sockets Layer (SSL) handshake.

WebSphere Application Server provides a default key manager that can select a certificate from a keystore when you define the following SSL configuration properties:

com.ibm.ssl.keyStoreClientAlias
: Defines the alias that is chosen from the keystore for the client side of a connection. This alias must be present in the keystore.

com.ibm.ssl.keyStoreServerAlias
: Defines the alias that is chosen from the keystore for the server side of a connection. This alias must be present in the keystore.

These two properties are set automatically when you use the administrative console because the default key manager is already configured.

With WebSphere Application Server, you can configure only one key manager at a time for a given SSL configuration. If you want custom certificate selection logic on the client side, you must write a new custom key manager. The custom key manager could provide function that prompts the user to choose a certificate dynamically. Also, you can implement an extended interface so that a key manager can provide information during connection time. For more information on the extended interface, see the com.ibm.wsspi.ssl.KeyManagerExtendedInfo interface. For more information on custom key manager development, see “Creating a custom key manager for SSL” on page 1766.

Default IbmX509 key manager

The default IbmX509 key manager chooses a certificate to serve as the identity for an SSL handshake. The key manager is called to enable client authentication on either side of the SSL handshake; frequently on the server-side, and less frequently on the client side according to client and server requirements. If a keystore is not configured on the client-side and SSL client authentication is enabled, the key manager cannot select a certificate to send to the server. Therefore, the handshake fails.

The following sample code shows the key manager configuration in the `security.xml` file for an IbmX509 key manager.

```
<keyManagers xmi:id="KeyManager_1" name="IbmX509"
provider="IBMJSSE2" algorithm="IbmX509" keyManagerClass=""
managementScope="ManagementScope_1"/>
```

You do not specify the keyManagerClass class because the key manager is provided by the IBMJSSE2 provider. However, you can specify whether the key manager is a custom class implementation, in which case you must specify the keyManager class, or an algorithm name that WebSphere Application Server can start from the Java security provider framework.

Custom key manager

The following sample code shows the key manager configuration in the `security.xml` file for a custom class.

```
<keyManagers xmi:id="KeyManager_2" name="CustomKeyManager"
keyManagerClass="com.ibm.ws.ssl.core.CustomKeyManager"
managementScope="ManagementScope_1"/>
```

The custom class must implement the javax.net.ssl.X509KeyManager interface and, optionally, implement the com.ibm.wsspi.ssl.KeyManagerExtendedInfo interface to retrieve additional WebSphere Application Server information. This interface replaces the function of the default key manager because you can configure only one key manager at a time. Therefore, the custom key manager has sole responsibility for selecting the alias to use from the configured keystore. The benefit of a custom key manager is its ability, on the client side, to prompt for an alias. This process enables the user to decide which certificate to use in situations where the user knows the client certificate identity. For more information, see “Creating a custom key manager for SSL” on page 1766.

Example: Enabling certificate revocation checking with the default IbmPKIX trust manager:

The IbmPKIX trust manager is enabled in the WebSphere Application Server by default. The IbmPKIX trust manager allows certificate revocation checking to occur. You enable certificate revocation checking by using the administrative console or by manually updating the `ssl.client.props` file.

The default IbmPKIX trust manager

The IbmPKIX trust manager is enabled by default, but revocation checking is not enabled by default. The following trust manager definition for IbmPKIX reflects the default condition:

```
<trustManagers xmi:id="TrustManager_managementNode_2" name="IbmPKIX" provider=
"IBMJSSE2" algorithm="IbmPKIX" trustManagerClass=""
managementScope="ManagementScope_managementNode_1">
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_1" name="com.ibm.se
curity.enableCRLDP" value="false" type="boolean" displayNameKey="" nlsRangeKey="
" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_2" name="com.ibm.js
se2.checkRevocation" value="false" type="boolean" displayNameKey="" nlsRangeKey=
"" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_3" name="ocsp.enable
e" value="false" type="String" displayNameKey="" nlsRangeKey="" hoverHelpKey=""
range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_4" name="ocsp.respo
nderURL" value="http://ocsp.example.net:80" type="String" displayNameKey=""
```

```
nlsRangeKey="" hoverHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_5" name="ocsp.respo
nderCertSubjectName" value="" type="String" displayNameKey="" nlsRangeKey="" hov
erHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_6" name="ocsp.respo
nderCertIssuerName" value="" type="String" displayNameKey="" nlsRangeKey="" hove
rHelpKey="" range="" inclusive="false" firstClass="false"/>
<additionalTrustManagerAttrs xmi:id="DescriptiveProperty_7" name="ocsp.respo
nderCertSerialNumber" value="" type="String" displayNameKey="" nlsRangeKey="" ho
verHelpKey="" range="" inclusive="false" firstClass="false"/>
</trustManagers>
```

Enabling certificate revocation checking with the default IbmPKIX trust manager

You can view and change IbmPKIX Trust Manager Custom Properties using the administrative console.

To do this,

- Click **Security > SSL certificate and key management**.
- Under Related Items, click **Trust managers**.
- Click **IbmPKIX**.
- Under Additional Properties, click **Custom properties**.

IbmPKIX custom properties

com.ibm.jsse2.checkRevocation

This property configures revocation checking for the Java Virtual Machine (JVM). This property is set to false by default because the default WebSphere certificates used for SSL communication do not contain certificate revocation list (CRL) distribution points or Online Certificate Status Protocol (OCSP) information.

Note: Since this property is a JVM property, this value is in effect for the entire application server. If the property is defined in trust managers at different scopes, the value in effect is used from the most specifically scoped IbmPKIX trust manager. For example, the property for an IbmPKIX trust manager defined at the node level overrides the property for an IbmPKIX trust manager defined at the cell level. This property is ignored for the IbmX509 trust manager.

default
false

com.ibm.security.enableCRLDP

This property configures CRL distribution point checking for the PKIX trust manager.

Note: If you enable CRL distribution point revocation checking, the certificates used for secure sockets layer (SSL) must contain a valid distribution point and the distribution point must be accessible or else SSL communication will fail and the server will not function correctly.

default
false

For certificates that do not contain an internal CRL distribution point, the following properties can used so the revocation status will be checked against a remote LDAP server containing the CRL.

com.ibm.security.ldap.certstore.host

This property specifies the LDAP server host name containing trusted certificates or certificate revocation lists. The target LDAP server host is used to obtain CA certificates or certificate revocation lists when validating a certificate and the local truststore does not contain the required certificate. The local truststore must contain the required certificates if an LDAP server is not

specified. In cases when an LDAP server is used, the root CA certificates must also be located in the local truststore as the LDAP server is not a trusted certificate store.

Note: Enabling this property in addition to the com.ibm.jsse2.checkRevocation property enables revocation checking. The remote LDAP server must contain a valid certificate revocation list and the server must be accessible. If the revocation status cannot be determined then the check will fail and SSL communication will fail and the server will not function correctly.

default
none

com.ibm.security.ldap.certstore.port
This property specifies the LDAP server port. A port value of 389 will be used by default if no LDAP server port is specified.

default
389

The following Java Development Kit (JDK) properties apply to enabling certificate revocation checking with the default IbmPKIX trust manager:

- ocsp.enable
- ocsp.responder
- ocsp.responderCertSubjectName
- ocsp.responderCertIssuerName
- ocsp.responderCertSerialNumber

These JDK properties can be set using the administrative console. You should reference Java(TM) Certification Path API Programmer's Guide - SDK 6.0 for descriptions of these properties and their allowable settings.

Note: In addition to its role of standard certificate verification, the IbmPKIX trust manager checks for certificates that contain CRL distribution points. This process is known as extended CRL checking. By default, CRL distribution point revocation checking is disabled. To enable CRL distribution point revocation checking, you must set the following properties to `true` using the administrative console:

- com.ibm.security.enableCRLDP
- com.ibm.jsse2.checkRevocation

OCSP properties and CRL properties affect certificate revocation checking. By default OCSP properties are checked first. If there is an error validating the certificate with OCSP, then validation uses a CRL distribution point instead.

When you select a trust manager, its associated properties are automatically set as Java system properties so that the IBMCertPath and IBMJSSE2 providers are aware that CRL checking is enabled or disabled. Similarly, the same applies for OCSP properties, which are java.security.Security properties.

Client considerations

You can also enable revocation checking for WebSphere application and administrative clients by directly setting the properties in the `ssl.client.props` file. An example of the `ssl.client.props` file follows:

```
#-------------------------------------------------------------------------
# Default Revocation Checking Properties
# These properties are used for certificate revocation checking with the IBM
# PKIX TrustManager.
#
# To enable CRL Distribution Points extension checking, use the system property
# com.ibm.security.enableCRLDP.
```

```
#
# OCSP checking is not enabled by default. It is enabled by setting the
# ocsp.enable property to "true".  Use of the other ocsp properties is optional.
#
# Note: Both OCSP and CRLDP checking is only effective if revocation checking
# has also been enabled by setting com.ibm.jsse2.checkRevocation to "true".
#
#-------------------------------------------------------------------------
com.ibm.jsse2.checkRevocation=false
com.ibm.security.enableCRLDP=false
#ocsp.enable=true
#ocsp.responderURL=http://ocsp.example.net
#ocsp.responderCertSubjectName=CN=OCSP Responder, O=XYZ Corp
#ocsp.responderCertIssuerName=CN=Enterprise CA, O=XYZ Corp
#ocsp.responderCertSerialNumber=2A:FF:00
```

Note: In order for these properties to be effective, you must ensure that the IbmPKIX trust manager is initialized by setting `com.ibm.ssl.trustManager=IbmPKIX`.

In addition, for revocation checking to be processed successfully on the client, you are required to turn off the signer exchange prompt. To do this, change the value of the **com.ibm.ssl.enableSignerExchangePrompt** property to `false`, in the ssl.client.props file.

For more information on these properties, see Java(TM) Certification Path API Programmer's Guide - SDK 6.0.

Keystore configurations for SSL

Use keystore configurations to define how the runtime for WebSphere Application Server loads and manages keystore types for Secure Sockets Layer (SSL) configurations.

By default, the java.security.Security.getAlgorithms("KeyStore") attribute does not display a predefined list of keystore types in the administrative console. Instead, WebSphere Application Server retrieves all of the KeyStore types that can be referenced by the java.security.KeyStore object, including hardware cryptographic, z/OS platform, IBM i platform, IBM Java Cryptography Extension (IBMJCE), and Java-based content management system (CMS)-provider keystores. If you specify a keystore provider in the `java.security` file or add it to the provider list programmatically, WebSphere Application Sever also retrieves custom keystores. The retrieval list depends upon the java.security configuration for that platform and process.

IBMJCE file-based keystores (JCEKS, JKS, and PKCS12)

A typical IBMJCE file-based keystore configuration is shown in the following sample code:

```
<keyStores xmi:id="KeyStore_1" name="NodeDefaultKeyStore"
password="{xor}349dkckdd=" provider="IBMJCE"
location="${USER_INSTALL_ROOT}/config/cells/myhostNode01Cell
/nodes/myhostNode01/key.p12" type="PKCS12" fileBased="true"
hostList="" initializeAtStartup="true" readOnly="false"
description="Default key store for myhostNode01" usage="SSLKeys"
managementScope="ManagementScope_1"/>
```

For more information about default keystore configurations, see “Default chained certificate configuration in SSL” on page 1724.

Table 1 describes the attributes that are used in the sample code.

Table 141. keystore configurations. This table describes the keystore configurations.

Attribute name	Default	Description
xmi:id	Varies	A value that issued to reference the keystore from another area in the configuration, for example, from an SSL configuration. Make this value unique within the `security.xml` file.

Table 141. keystore configurations (continued). This table describes the keystore configurations.

Attribute name	Default	Description
name	For Java Secure Socket Extension (JSSE) keystore: NodeDefaultKeyStore. For JSSE truststore: NodeDefaultTrustStore.	A name that is used to identify the keystore by sight. The name can determine if the keystore is a default keystore based upon whether the name ends with DefaultKeyStore or DefaultTrustStore.
password	The default keystore password is `WebAS`. It is recommended that this be changed as soon as possible. See Updating default key store passwords using scripting for more information.	The password that is used to access the keystore name is also the default that is used to store keys within the keystore.
description	No default	A description of the keystore.
usage	An attribute specifying what the keystore is used for.	Valid values are: SSLKeys, KeySetKeys, RootKeys, DeletedKeys, DefaultSigners, RSATokenKeys.
provider	The default provider is IBMJCE.	The Java provider that implements the type attribute (for example, PKCS12 type). The provider can be left unspecified and the first provider that implements the keystore type specified is used.
location	The default varies, but typically references a `key.p12` file or a `trust.p12` file in the node or cell directories of the configuration repository. These files are PKCS12 type keystores.	The keystore location reference. If the keystore is file-based, the location can reference any path in the file system of the node where the keystore is located. However, if the location is outside of the configuration repository, and you want to manage the keystore remotely from the administrative console or from the **wsadmin** utility, then specify the hostList attribute that contains the host name of the node where it resides.
type	The default Java crypto device keystore type is `PKCS12`.	This type specifies the keystore. Valid types can be those returned by the java.security.Security.getAlgorithms("KeyStore") attribute. These types include the following keystore types, and availability depends on the process and platform `java.security` configuration: • JKS • JCEKS • PKCS12 • PKCS11 (Java crypto device) • CMSKS • IBMi5OSKeyStore • JCERACFKS • JCECCAKS keystores (replacing JCE4758KS) - (z/OS crypto device)
fileBased	The default is `true`.	This option is required for default keystores. It indicates a file-system keystore so you can use a FileInputStream or FileOutputStream for loading and storing the keystore.
hostList	The hostList attribute is used to specify a remote hostname so that the keystore can be remotely managed. There are no remotely managed keystores by default. All default keystores are managed locally in the configuration repository and synchronized out to each of the nodes.	The option manages a keystore remotely. You can set the host name of a valid node for a keystore. When you use either the administrative console or the wsadmin utility to manage certificates for this keystore, an MBean call is made to the node where the keystore exists for the approved operation. You can specify multiple hosts, although synchronization of the keystore operations are not guaranteed. For example, one of the hosts that is listed might be down when a specific operation is performed. Therefore, use multiple hosts in this list.
initializeAtStartup	The default is `true`.	This option informs the runtime to initialize the keystore during startup. This option can be important for hardware cryptographic device acceleration.
readOnly	The default is `false`.	This option informs the configuration that you cannot write to this keystore. That is, certain update operations on the keystore cannot be attempted and are not allowed. An example of a read-only keystore type is JCERACFKS on the z/OS platform. This type is read-only from the WebSphere certificate management standpoint, but you can also update it using the keystore management facility for RACF.
managementScope	The default scope is the node scope for a base Application Server environment and the cell scope for a Network Deployment environment.	This option references a particular management scope in which you can see this keystore. For example, if a hardware cryptographic device is physically located on a specific node, then create the keystore from a link to that node in the topology view under **Security > Security Communications > SSL configurations**. You can also use management scope to isolate a keystore reference. In some cases, you might need to allow only a specific application server to reference the keystore; the management scope is for that specific server.

CMS keystores

You can set some provider-specific attributes in CMS keystores.

When you create a CMS keystore, the CMS provider is `IBMi5OSJSSEProvider`, and the CMS type is `IBMi5OSKeyStore`, as shown in the following sample code:

```
<keyStores xmi:id="KeyStore_1132071489571" name="CMSKeyStore"
password="{xor}HRYNFAtrbxEwOzpvbhw6MzM=" provider="IBMi5OSJSSEProvider"
location="${USER_INSTALL_ROOT}\profiles\AppSrv01/config/cells/myhostCell01
/nodes/myhostNode01/servers/webserver1/plugin-key.kdb" type="IBMi5OSKeyStore"
fileBased="true" createStashFileForCMS="true"
managementScope="ManagementScope_1132071489569"/>
```

Note: The IBM i keystore type `IBMi5OSKeyStore` does not recognize or generate `.sth` password stash files. Instead it keeps an internal record of the password for the `.kdb` keystore file where it is created. If the `.kdb` file is moved, the password is no longer associated with the keystore. In that case, the Digital Certificate Manager (DCM) must be used to recreate the internal record of the password for the `.kdb` key store file. For more information, see “Recreating the .kdb keystore internal password record” on page 1793.

Attention: When you create chained personal certificates or use the requestCACertificate task with the IBMi5OSKeyStore, the IBMi5OSJSSEProvider requires that the signer for each part of the chain be present in the keystore prior to creation of the new certificate. Therefore, you must import the signer into the IBMi5OSKeyStore keystore before creating the new certificate.

Hardware cryptographic keystores

For cryptographic device configuration, see “Key management for cryptographic uses” on page 1834.

You can add a slot either as the custom property, com.ibm.ssl.keyStoreSlot, or as the configuration attribute, `slot="0"`. The custom property is read before the attribute for backwards compatibility.

Dynamic outbound selection of Secure Sockets Layer configurations

WebSphere Application Server provides dynamic outbound selection that enables you to choose a specific Secure Sockets Layer (SSL) configuration and certificate alias for each outbound protocol, target host, target port, or any combination of these attributes. You can specify the dynamic selection information for outbound connections from a pure client or from a server that is acting as a client.

Before the SSL runtime for WebSphere Application Server starts an outbound connection, the runtime attempts to match the outbound protocol, target host, and target port attributes with the dynamic outbound selection information that is associated with an SSL configuration and certificate alias in the configuration.

The runtime caches both selection misses and selection hits, so the impact on performance can be minimal. However, a relationship exists between the amount of dynamic outbound selection information and its impact on the initial connection performance.

Target information during outbound connections

The dynamic outbound selection configurations are only effective when the outbound protocol uses the JSSEHelper application programming interface (API) when you select an SSL configuration with a specified connectionInfo hash map. This hash map must contain the following properties:

com.ibm.ssl.direction
The value for outbound connections is OUTBOUND.

com.ibm.ssl.remoteHost
The format should match what the protocol provides. Typically this is the canonical Domain Name Space (DNS), but it also could be the IP address.

com.ibm.ssl.remotePort
The port is target port.

com.ibm.ssl.endPointName
The value for an outbound connection must be one of the following protocol strings:

- IIOP
- HTTP
- SIP
- LDAP
- ADMIN_IPC
- ADMIN_SOAP
- BUS_TO_BUS
- BUS_CLIENT
- BUS_TO_WEBSPHERE_MQ
- WEBSPHERE_MQ_CLIENT

Central management of SSL configurations

By default, Secure Sockets Layer (SSL) configurations for servers are managed from a central location in the topology view of the administrative console. You can associate an SSL configuration and certificate alias with a specific management scope. This method is the most efficient method to manipulate and modify configurations when the server topology changes.

In prior releases, SSL configurations are managed for each process. You have to maintain individual settings for each SSL configuration in the topology. In this release of WebSphere Application Server, management control of your SSL configurations offers more options and additional flexibility. You are able to make coarse-grained changes for the entire topology using the cell-scope and also make fine-grained changes using a particular endpoint name for a specific application server process. Because the SSL configuration associations manifest an inheritance behavior, you can simplify the number of associations by referencing only the highest level management scope that needs a unique configuration.

The topology view provides the scoping mechanism. The SSL configuration inherits its scope, which can be seen as its display in the topology. The scope encompasses the level where you created the configuration and all the levels below that point. For example, when you create an SSL configuration at a specific node, that configuration can be seen by that node agent and by every application server that is part of that node. Any application server or node that is not part of this particular node can not see this SSL configuration.

Your security environment influences issues such as the uniqueness of the SSL configurations, as well as the SSL configuration and the certificate alias placement in the topology. You are also able to configure different certificate aliases and different SSL configurations for inbound connections versus outbound connections.

To configure the inbound and outbound topologies, which must be done separately in the administrative console, click **Security > SSL certificates and key management > Manage endpoint security configurations > Inbound | Outbound**.

Default centrally managed SSL configuration

The default management scope is the node scope. When a node is federated into a cell, the default SSL configurations for the node are maintained, as shown in the following sample code for the sslConfigGroups and management scopes attributes:

```
<sslConfigGroups xmi:id="SSLConfigGroup_1" name="myhostNode01"
direction="inbound" certificateAlias="default" sslConfig="SSLConfig_1"
managementScope="ManagementScope_1"/>
```

```
<sslConfigGroups xmi:id="SSLConfigGroup_2" name="myhostNode01"
direction="outbound" certificateAlias="default" sslConfig="SSLConfig_1"
managementScope="ManagementScope_1"/>

<managementScopes xmi:id="ManagementScope_1"
scopeName="(cell):myhostNode01Cell:(node):myhostNode01" scopeType="node"/>
```

The SSL configuration xmi:id "SSLConfig_1" is also federated and applicable:

```
<repertoire xmi:id="SSLConfig_1" alias="NodeDefaultSSLSettings"
managementScope="ManagementScope_1">
<setting xmi:id="SecureSocketLayer_1" clientAuthentication="true"
securityLevel="HIGH" enabledCiphers="" jsseProvider="IBMJSSE2"
sslProtocol="SSL_TLS" keyStore="KeyStore_1" trustStore="KeyStore_2"
trustManager="TrustManager_1" keyManager="KeyManager_1"/>
</repertoire>
```

The keystores that are associated with the SSLConfig_1 SSL configuration are also federated, and `key.p12` is located in the node directory of the configuration repository:

```
<keyStores xmi:id="KeyStore_1" name="NodeDefaultKeyStore"
password="{xor}HRYNFAtrbxEwOzpvbhw6MzM=" provider="IBMJCE"
location="${USER_INSTALL_ROOT}/config/cells/myhostNode01Cell/nodes
/myhostNode01/key.p12" type="PKCS12" fileBased="true" hostList=""
initializeAtStartup="true" managementScope="ManagementScope_1"/>
<keyStores xmi:id="KeyStore_2" name="NodeDefaultTrustStore"
password="{xor}HRYNFAtrbxEwOzpvbhw6MzM=" provider="IBMJCE"
location="${USER_INSTALL_ROOT}/config/cells/myhostNode01Cell
/nodes/myhostNode01/trust.p12" type="PKCS12" fileBased="true"
hostList="" initializeAtStartup="true" managementScope="ManagementScope_1"/>
```

Secure Sockets Layer node, application server, and cluster isolation

Secure Sockets Layer (SSL) enables you to ensure that any client that attempts to connect to a server during the handshake first performs server authentication. Using SSL configurations at the node, application server, and cluster scopes, you can isolate communication between severs that should not be allowed to communicate with each other over secure ports.

Before you attempt to isolate communications controlled by WebSphere Application Server, you must have a good understanding of the deployment topology and application environment. To isolate a node, application server, or cluster, you must be able to control the signers that are contained in the truststores that are associated with the SSL configuration. When the client does not contain the server signer, it cannot establish a connection to the server. By default, WebSphere uses chained certificates and each node has a unique root certificate signer. Because they the node shares the same root signer, all of the server in that node can connect to each other because they share the same root signer. However, if you use self-signed certificates, the server that created the personal certificate controls the signer, although you do have to manage the self-signed certificates. If you obtain certificates from a certificate authority (CA), you must obtain multiple CA signers because all of the servers can connect to each other if they share the same signer.

Authenticating only the server-side of a connection is not adequate protection when you need to isolate a server. Any client can obtain a signer certificate for the server and add it to its trust store. SSL client authentication must also be enabled between servers so that the server can control its connections by deciding which client certificates it can trust. For more information, see “Enabling Secure Sockets Layer client authentication for a specific inbound endpoint” on page 1777, which applies as well to enabling SSL client authentication at the cell level.

Isolation also requires that you use centrally managed SSL configurations for all or most endpoints in the cell. Centrally managed configurations can be scoped, unlike direct or end point configuration selection, and they enable you to create SSL configurations, key stores, and trust stores at a particular scope. Because of the inheritance hierarchy of WebSphere Application Server cells, if you select only the properties that you need for an SSL configuration, only these properties are defined at your selected

scope or lower. For example, if you configure at the node scope, your configuration applies to the application server and individual end point scopes below the node scope. For more information, see “Associating Secure Sockets Layer configurations centrally with inbound and outbound scopes” on page 1774, “Selecting an SSL configuration alias directly from an endpoint configuration” on page 1775, and “Associating a Secure Sockets Layer configuration dynamically with an outbound protocol and remote secure endpoint” on page 1771

When you configure the key stores, which contain cryptographic keys, you must work at the same scope at which you define the SSL configuration and not at a higher scope. For example, if you create a key store that contains a certificate whose host name is part of the distinguished name (DN), then store that keystore in the node directory of the configuration repository. If you decide to create a certificate for the application server, then store that keystore on the application server in the application server directory.

When you configure the trust stores, which control trust decisions on the server, you must consider how much you want to isolate the application servers. You cannot isolate the application servers from the node agents or the deployment manager. However, you can configure the SOAP connector end points with the same personal certificate or to share trust. Naming persistence requires IIOP connections when they pass through the deployment manager. Because application servers always connect to the node agents when the server starts, the IIOP protocol requires that WebSphere Application Server establish trust between the application servers and the node agents.

Establishing node SSL isolation

By default, WebSphere Application Server installation uses a single chained certificate for each node so you can isolate nodes easily. A common trust store, which is located in the cell directory of the configuration repository, contains all of the signers for each node that is federated into the cell. After federation, each cell process trusts all of the other cell processes because every SSL configuration references the common trust store.

You can modify the default configuration so that each node has its own trust store, and every application server on the node trusts only the node agent that uses the same personal certificate. You must also add the signer to the node trust store so that WebSphere Application Server can establish trust with the deployment manager. To isolate the node, ensure that the following conditions are met:

- The deployment manager must initiate connections to any process
- The node agent must initiate connections to the deployment manager and its own application servers
- The application servers must initiate connections to the applications servers on the same node, to its own node agent, and the deployment manager

Figure 1 shows Node Agent A contains a key.p12 keystore and a trust.p12 trust store at the node level of the configuration repository for node A.

Figure 32.

When you associate an SSL configuration with this keystore and truststore, you break the link with the cell-scoped trust store. To isolate the node completely, repeat this process for each node in the cell. WebSphere Application Server SSL configurations override the cell scope and use the node scope instead so that each process at this scope uses the SSL configuration and certificate alias that you selected at this scope. You establish proper administrative trust by ensuring that nodeA signer is in the common trust store and the cell signer is in the nodeA trust store. The same logic applies to node B as well. For more information, see “Associating Secure Sockets Layer configurations centrally with inbound and outbound scopes” on page 1774.

Establishing application server SSL isolation

Isolating application server processes from one another is more challenging than isolating nodes. You must consider the following application design and topology conditions:

- An application server process might need to communicate with the node agent and deployment manager
- Isolating application server processes from each other might disable single sign-on capabilities for horizontal propagation

If you configure outbound SSL configurations dynamically, you can accommodate these conditions. When you define a specific outbound protocol, target host, and port for each different SSL configuration, you can override the scoped configuration.

Figure 2 shows how you might isolate an application server completely, although in practice this approach would be more complicated.

Figure 33.

The dynamic configuration enables server1 on Node A to communicate with server 1 on Node B only over IIOP. The dynamic outbound rule is IIOP,nodeBhostname,*. For more information, see “Associating a Secure Sockets Layer configuration dynamically with an outbound protocol and remote secure endpoint” on page 1771

Establishing cluster SSL isolation

You can configure application servers into clusters instead of scoping them centrally at the node or dynamically at the server to establish cluster SSL isolation. While clustered servers can communicate with each other, application servers outside of the cluster cannot communicate with the cluster, thus isolating the clustered servers. For example, you might need to separate applications from different departments while maintaining a basic level of trust among the clustered servers. Using the dynamic outbound SSL configuration method described for servers above, you can easily extend the isolated cluster as needed.

Figure 3 shows a sample cluster configuration where cluster 1 contains a key.p12 with its own self-signed certificate, and a trust.p12 that is located in the config/cells/<cellname>/clusters/<clustername> directory.

Figure 34.

In the example, cluster1 might contain web applications, and cluster2 might contain EJB applications. Considering the various protocols, you decide to enable IIOP traffic between the two clusters. Your task is to define a dynamic outbound SSL configuration at the cluster1 scope with the following properties:

```
IIOP,nodeAhostname,9403|IIOP,nodeAhostname,9404|IIOP,nodeBhostname,9403|IIOP,nodeBhostname,9404
```

You must create another SSL configuration at the cluster1 scope that contains a new trust.p12 file with the cluster2 signer. Consequently, outbound IIOP requests go either to nodeAhostname ports 9403 and 9404 or to nodeBhostname ports 9403 and 9404. The IIOP SSL port numbers on these two application server processes in cluster2 identify the ports.

As you review Figure 3, notice the following features of the cluster isolation configuration:

- The trust.p12 for cluster1 contains signers that allow communications with itself (cluster1 signer), between both node agents (nodeAsigner and nodeBsigner), and with the deployment manager (cell signer).
- The trust.p12 for cluster2 contains signers that allow communications with itself (cluster2 signer), between both node agents (nodeAsigner and nodeBsigner), and with the deployment manager (cell signer).

- Node agent A and Node agent B can communicate with themselves, the deployment manager, and both clusters.

For more information, see “Associating a Secure Sockets Layer configuration dynamically with an outbound protocol and remote secure endpoint” on page 1771.

Although this article presents an overview of isolation methods from an SSL perspective, you must also ensure that non-SSL ports are closed or applications require the confidentiality constraint in the deployment descriptor. For example, you can set the CSIv2 inbound transport panel to require SSL and disable the channel ports that are not secure from the server ports configuration.

Also, you must enable SSL client authentication for SSL to enforce the isolation requirements on both sides of a connection. Without mutual SSL client authentication, a client can easily obtain a signer for the server programmatically and thus bypass the goal of isolation. With SSL client authentication, the server would require the client's signer for the connection to succeed. For HTTP/S protocol, the client is typically a browser, a web service, or a URL connection. For the IIOP/S protocol, the client is typically another application server or a Java client. WebSphere Application Server must know the clients to determine if SSL client authentication enablement is possible. Any applications that are available through a public protocol must not enable SSL client authentication because the client may fail to obtain a certificate to authenticate to the server.

Note: It is beyond the scope of this article to describe all of the factors you must consider to achieve complete isolation.

Certificate options during profile creation

Starting in WebSphere Application Server Version 7.0, you have several options available during profile creation concerning the default certificate and root certificate of the server.

The new certificate options enable you to:

- Import the default certificate of the server
- Import the root certificate of the server
- Customize the default certificate subjectDN and validity period of the server
- Customize the root certificate subjectDN and validity period of the server

Two new panels are available during profile creation that enable you to make decisions about the default certificate and root certificate of the server.

The first panel, titled Security Certificate (Part 1), enables you to choose to import a certificate or to have WebSphere Application Server create the default certificate or the default root certificate of the server for you.

The second panel, titled Security Certificate (Part 2), either displays the information from the certificate imported from the previous panel, or, if you choose to have WebSphere Application Server create the certificate, enables you to change the subjectDN and the certificate validity period.

Customization of certificates can also be performed by using the manageprofile command and from a silent install response file.

Importing the default certificate of the server during profile creation

If the default certificate of the server is imported during profile creation, it is added to NodeDefaultKeyStore if on a stand-alone application server, or to CellDefaultKeyStore if on a deployment manager. The imported certificate signer is added to NodeDefaultTrustStore or CellDefaultTrustStore.

To import the default certificate of the server, you must have a personal certificate stored and a keystore that you have access to. You must know the location, type and password of the keystore. On the Security Certificate (Part 1) panel, do the following:

1. Select **Import an existing default personal certificate**.
2. Type or select the keystore file name.
3. Enter the password of the keystore.
4. Select a keystore type from the pull-down list.
5. If you have correctly filled in all information from the previous 3 steps, you are able to select a certificate alias from the pull-down list.

The certificate you choose is imported to the default keystore of the server. The next panel, Security Certificate (Part 2) displays the issuedTo and issuedBy certificate information.

If you use the manageprofiles command to import the default certificate, the options are:

-importPersonalCertKS keystore_path
the keystore file location

-importPersonalCertKSType keystore_type
the type of the keystore

-importPersonalCertKSPassword keystore_password
the password to open the keystore

-importPersonalCertKSAlias keystore_alias
the alias of the certificate used from the keystore

Importing the root certificate of the server during profile creation

If the server root certificate is imported during profile creation, the certificate is added to NodeDefaultRootStore on a stand-alone application server or to DmgrDefaultRootStore on a deployment manager. The signer is pulled from the imported root certificate and added to NodeDefaultTrustStore or CellDefaultTrustStore. The root certificate is used by WebSphere Application Server to sign any chained certificates it creates. If no default certificate is provided during profile creation, WebSphere Application Server uses the root certificate to sign the default certificate of the server.

To import the default certificate of the server, you must have a personal certificate stored and a keystore that you have access to. You must know the location, type and password of the keystore. On the Security Certificate (Part 1) panel, do the following:

1. Select **Import an existing root signing certificate**.
2. Type or select the keystore file name.
3. Enter the password of the keystore.
4. Select a keystore type from the pull-down list.
5. If you have correctly filled in all information from the previous 3 steps, you are able to select a certificate alias from the pull-down list.

The certificate you choose is imported to the root keystore of the server. The next panel, Security Certificate (Part 2) displays the issuedTo and issuedBy certificate information.

If you use the manageprofiles command to import the root certificate, the options are:

-importSigningICertKS keystore_path
the keystore file location

-importSigningCertKSType keystore_type
the type of the keystore

-importSigningCertKSPassword keystore_password
the password to open the keystore

-importSigningCertKSAlias keystore_alias
the alias of the certificate used from the keystore

Customizing the default certificate created by WebSphere Application Server

If you choose to let WebSphere Application Server create the default certificate of the server, you can customize the subject distinguished name (DN) and the life span of the certificate.

To customize the default certificate of the server on the Security Certificate (Part 1) panel, do the following:

1. Select **Create a new default personal certificate**.
2. On the next panel, Security Certificate (Part 2), the Issued to distinguished name field contains the WebSphere Application Server default DN. Replace this with your customized DN.
3. In Expiration period in years, select the number of years you want the certificate to be valid for.

If you use the manageprofiles command to customize the default certificate, the options are:

-personalCertDN distinguished_name
the DN to give to the certificate

-personalCertValidityPeriod validity_period
the life span to give to the certificate

Customizing the root certificate created by WebSphere Application Server

If you choose to let WebSphere Application Server create the root certificate, you can customize the DN of the certificate and the life span of the certificate.

To customize the root certificate of the server on the Security Certificate (Part 1) panel, do the following:

1. Select **Create a new root signing certificate**.
2. On the next panel, Security Certificate (Part 2), the Issued by distinguished name field contains the WebSphere Application Server default root certificate DN. Replace this with your customized DN.
3. In Expiration period in years, select the number of years you want the root certificate to be valid for.

If you use the manageprofiles command to customize the root certificate, the options are:

-signingCertDN distinguished_name
the DN to give to the root certificate

-signingCertValidityPeriod validity_period
the life span to give to the root certificate

Default chained certificate configuration in SSL

When a WebSphere Application Server process starts for the first time, the Secure Sockets Layer (SSL) runtime initializes the default keystores and truststores that are specified in the SSL configuration.

The chained certificates created during profile creation have a 1 year life span by default. The default root certificate used to signer the default chained certificate has a life span of 15 years. The life span of the default and the root certificates can be customized during profile creation. An advantage in this type of chained certificate is that only the signer from the root certificate is needed to establish trust. When the chained certificate is regenerated with the same root certificate, clients using that root signer certificate for trust do not lose their trust.

Default keystore and truststore properties
WebSphere Application Server creates the `key.p12` default keystore file and the `trust.p12` default truststore file during profile creation. A default, chained certificate is also created in the **key.p12**

file. The root signer, or public key, of the chained certificate is extracted from the **key.p12** file and added to the **trust.p12** file. If the files do not exist during process startup, they are recreated during startup.

You can identify keystore and truststore defaults because of their suffixes: **DefaultKeyStore** and **DefaultTrustStore**. Also, in the SSL configuration, you must set the **fileBased** attribute to `true` so that the runtime only uses the default keystores and truststore.

On a base application server, default key and truststores are stored in the node directory of the configuration repository. For example, the default `key.p12` and `trust.p12` stores are created with the AppSrv01 profile name, the myhostNode01Cell name, and the myhostNode01 node name. The keystore and truststore are located in the following directories:

- C:\WebSphere\AppServer\profiles\AppSrv01\config\cells\myhostNode01Cell \nodes\myhostNode01\key.p12
- C:\WebSphere\AppServer\profiles\AppSrv01\config\cells\myhostNode01Cell \nodes\myhostNode01\trust.p12

The default password is **WebAS** for all default keystores generated by WebSphere Application Server. Change the default password after the initial configuration for a more secure environment.

Default chained certificate

The default chained certificate of the server along with a root self-signed certificate used to sign the default chained certificate are created during profile creation.

You can recreate the certificates with different information simply by deleting the `*.p12` files in `/config` and `/etc`. Change the four properties in the next code example to the values you want the certificates to contain, then restart the processes. This causes the server certificate in `/config` and the client certificate in `/etc` to differ.

The certificate properties in the next code example exist in the **ssl.client.props** file, but do not exist in the server configuration. However, you can use these values in the server configuration by adding them as custom security properties in the administrative console. Click **Security** > **Global securit**y > **Custom properties** to change the following properties:

```
com.ibm.ssl.defaultCertReqAlias=default_alias
com.ibm.ssl.defaultCertReqSubjectDN=cn=${hostname},ou=myhostNode01,ou=myhostNode01Cell,o=IBM,c=US
com.ibm.ssl.defaultCertReqDays=365
com.ibm.ssl.defaultCertReqKeySize=1024
com.ibm.ssl.rootCertSubjectDN=cn=${hostname},ou=Root Certificate, ou=myhostNode01,
ou=myhostNode01Cell,o=IBM,c=US
com.ibm.ssl.rootCertValidDays=7300
com.ibm.ssl.rootCertAlias=root
com.ibm.ssl.rootCertKeySize=1024
```

After changing the properties, complete the following actions:

1. Delete the default key.p12 keystore and trust.p12 truststore files for the application server, which contain the default chained certificate. If the keystore and truststore file do not exist, WebSphere Application Server automatically generates them and creates new default certificates using the previously listed property values.
2. Delete the root keystore, which is the root-key.p12 file, to regenerate the root certificate with the previously listed properties.
3. Restart the application server.

If a `default_alias` value already exists, the runtime appends `_#`, where the number sign (#) is a number that increases until it is unique in the keystore. `${hostname}` is a variable that is resolved to the host name where it was originally created. The default expiration date of chained certificates is one year from their creation date.

The runtime monitors the expiration dates of chained certificates using the certificate expiration monitor. These chained certificates are automatically replaced along with any signer certificates when they are within the expiration threshold, which is typically 30 days before expiration. You can

increase the default key size beyond 1024 bits only when the Java runtime environment policy files are unrestricted (that is, not exported). For more information, see “Certificate expiration monitoring in SSL” on page 1732.

Default keystore and truststore configurations for new Base Application Server processes

The following sample code shows the default SSL configuration for a base application server. References to the default keystores and truststores files are highlighted.

```
<repertoire xmi:id="SSLConfig_1" alias="NodeDefaultSSLSettings"
managementScope="ManagementScope_1">
<setting xmi:id="SecureSocketLayer_1" clientAuthentication="false"
securityLevel="HIGH" enabledCiphers="" jsseProvider="IBMJSSE2" sslProtocol="SSL_TLS"
keyStore="KeyStore_1" trustStore="KeyStore_2" trustManager="TrustManager_1"
keyManager="KeyManager_1"/>
</repertoire>
```

Default keystore

In the following sample code, the keystore object that represents the default keystore is similar to the XML object.

```
<keyStores xmi:id="KeyStore_1" name="NodeDefaultKeyStore"
password="{xor}349dkckdd=" provider="IBMJCE" location="${WAS_INSTALL_ROOT}/config
/cells/myhostNode01Cell/nodes/myhostNode01/key.p12" type="PKCS12" fileBased="true"
hostList="" initializeAtStartup="true" managementScope="ManagementScope_1"/>
```

The **NodeDefaultKeyStore** keystore contains the personal certificate that represents the identity of the secure endpoint. Any keystore reference can use the ${WAS_INSTALL_ROOT} variable, which is expanded by the runtime. The PKCS12 default keystore type is in the most interoperable format, which means that it can be imported into most browsers. The myhostNode01Cell password is encoded. The management scope determines which server runtime loads the keystore configuration into memory, as shown in the following code sample:

```
<managementScopes xmi:id="ManagementScope_1" scopeName="
(cell):myhostNode01Cell:(node):myhostNode01" scopeType="node"/>
```

Any configuration objects that are stored in the **security.xml** file whose management scopes are outside the current process scope are not loaded in the current process. Instead, the management scope is loaded by servers that are contained within the myhostNode01 node. Any application server that is on the specific node can view the keystore configuration.

When you list the contents of the **key.p12** file to show the chained certificate, note that the common name (CN) of the distinguished name (DN) is the host name of the resident machine. This listing enables you to verify the host name by its URL connections. Additionally, you can verify the host name from a custom trust manager. For more information, see “Trust manager control of X.509 certificate trust decisions” on page 1709.

Contents of default keystore

The following sample code shows the contents of the default **key.p12** file in a keytool list:

```
keytool -list -v -keystore c:\WebSphere\AppServer\profile\AppSrv01\profiles\config
\cells\myhostNode01Cell\nodes\myhostNode01\key.p12 -storetype PKCS12 -storepass *****

Keystore type: PKCS12
Keystore provider: IBMJCE

Your keystore contains 1 entry

Alias name: default
Creation date: Dec 31, 1969
Entry type: keyEntry
Certificate chain length: 2
Certificate[1]:
Owner: CN=myhost.austin.ibm.com, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US
Issuer: CN=myhost.austin.ibm.com, OU=Root Certificate, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US
Serial number: 4e48f29aafea6
Valid from: 2/7/08 1:03 PM until: 2/6/09 1:03 PM
Certificate fingerprints:
  MD5:  DB:FE:65:DB:40:13:F4:48:A4:CE:2F:4F:60:A5:FF:2C
  SHA1: A1:D4:DD:4B:DE:7B:45:F7:4D:AA:6A:FC:92:38:78:53:7A:99:F1:DC
Certificate[2]:
Owner: CN=myhost.austin.ibm.com, OU=Root Certificate, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US
Issuer: CN=myhost.austin.ibm.com, OU=Root Certificate, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US
Serial number: 4e48e5fd4eae3
Valid from: 2/7/08 1:03 PM until: 2/2/28 1:03 PM
Certificate fingerprints:
  MD5:  A5:9B:05:78:CF:AB:89:94:C9:2E:F1:87:34:B3:FC:75
```

```
  SHA1: 43:74:B6:C7:FA:C1:0F:19:F2:51:2B:17:60:0D:34:93:55:BF:D5:D2

*******************************************
*******************************************
```

The default alias name and the keyEntry entry type indicate that the private key is stored with the public key, which represents a complete personal certificate. The certificate is owned by`CN=myhost.austin.ibm.com, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US` and it is issued by the default root certificate, which is owned by`CN=myhost.austin.ibm.com, OU=Root Certificate, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US` By default, the certificate is valid for one year from the date of creation.

Additionally, in some signer-exchange situations, the certificate fingerprint ensures that the sent certificate has not been modified. The fingerprint, which is a hash algorithm output for the certificate, is displayed by the WebSphere Application Server runtime during an automated signer exchange on the client side. The client fingerprint must match the fingerprint that is displayed on the server. The runtime typically uses the SHA1 hash algorithm to generate certificate fingerprints.

Default truststore

In the following sample code, the keystore object represents the default `trust.p12` truststore. The truststore contains signer certificates that are necessary for making trust decisions:

```
<keyStores xmi:id="KeyStore_2" name="NodeDefaultTrustStore"
password="{xor}349dkckdd=" provider="IBMJCE" location="${WAS_INSTALL_ROOT}
/config/cells/myhostNode01Cell/nodes/myhostNode01/trust.p12" type="PKCS12"
fileBased="true" hostList="" initializeAtStartup="true" managementScope="ManagementScope_1"/>
```

Contents of default truststore

The following sample code shows the contents of the default `trust.p12` truststore in a keytool listing. By default, for the sample chained certificate, the root certificate signer is included in the trust store. The root signer alias name and the trustedCertEntry entry type indicate that the certificate is the public key. The private key is not stored in this truststore. In addition, all truststores contain the default DataPower certificate.

```
keytool -list -v -keystore c:\WebSphere\AppServer\profile\AppSrv01\profiles\config\cells\myhostNode01Cell
\nodes\myhostNode01\trust.p12 -storetype PKCS12 -storepass *****
```

```
Keystore type: PKCS12
Keystore provider: IBMJCE

Your keystore contains 2 entries

Alias name: root
Creation date: Dec 31, 1969
Entry type: trustedCertEntry

Owner: CN=myhost.austin.ibm.com, OU=Root Certificate, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US
Issuer: CN=myhost.austin.ibm.com, OU=Root Certificate, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US
Serial number: 4e48e5fd4eae3
Valid from: 2/7/08 1:03 PM until: 2/2/28 1:03 PM
Certificate fingerprints:
   MD5:  A5:9B:05:78:CF:AB:89:94:C9:2E:F1:87:34:B3:FC:75
   SHA1: 43:74:B6:C7:FA:C1:0F:19:F2:51:2B:17:60:0D:34:93:55:BF:D5:D2

*******************************************
*******************************************

Alias name: datapower
Creation date: Dec 31, 1969
Entry type: trustedCertEntry

Owner: OU=Root CA, O="DataPower Technology, Inc.", C=US
Issuer: OU=Root CA, O="DataPower Technology, Inc.", C=US
Serial number: 0
Valid from: 6/11/03 1:23 PM until: 6/6/23 1:23 PM
Certificate fingerprints:
   MD5:  18:AC:86:D1:9A:90:A2:AE:8B:28:F9:A8:75:C8:A9:DB
   SHA1: A9:BA:A4:B5:BC:26:2F:5D:2A:80:93:CA:BA:F4:31:05:F2:54:14:17
```

Secure installation for client signer retrieval in SSL:

Each profile in the WebSphere Application Server environment contains a unique chained certificate signed by a unique long lived root certificate that was created when the profile was created. This certificate replaces the default self-signed certificate that ships with WebSphere Application Server Version 6.1 as

well as the default dummy certificate that ships in releases prior to Version 6.1. When a profile is federated to a deployment manager, the signer for the root signing certificate is added to the common truststore for the cell, establishing trust for all certificates signed by that root certificate.

Note: Do not use the dummy keystore and truststore files, which are referenced in this topic, in a production environment. These files contain the same certificates and are used everywhere, which is not secure. Also, change the passwords for the keystore and truststore so that it does not use the `WebAS` default password.

By default, clients do not trust servers from different profiles in the WebSphere Application Server environment. That is, they do not contain the root signer for these servers. There are some things that you can do to assist in establishing this trust:

1. Enable the signer exchange prompt to except the signer during the connection attempt.
2. Run the **retrieveSigners** utility to download the signers from that system prior to making the connection.
3. Copy the `trust.p12` file from the `/config/cells/<cell_name>/nodes/<node_name>` directory of the server profile to the `/etc` directory of the client. Update the SSL configuration to reflect the new file name and password, if they are different. Copying the file provides the client with a `trust.p12` that contains all signers from servers in that cell. Also, you might need to perform this step for back-level clients that are still using the `DummyClientTrustFile.jks` file. In this case, you might need to change the `sas.client.props` or `soap.client.props` file to reflect the new truststore, truststore password, and truststore type (`PKCS12`).

For clients to perform an in-band signer exchange, you must specify the `ssl.client.props` file as a com.ibm.SSL.ConfigURL property in the SSL configuration. For managed clients, this is done automatically. Signers are designated either as in-band during the connection or out-of-band during runtime. You must also set the com.ibm.ssl.enableSignerExchangePrompt attribute to `true`.

Tip: You can configure a certificate expiration monitor to replace server certificates that are about to expire. For more information about how clients can retrieve the new signer from the configuration, see “Certificate expiration monitoring in SSL” on page 1732.

Using the signer exchange prompt to retrieve signers from a client

When the client does not already have a signer to connect to a process, you can enable the signer exchange prompt. The signer exchange prompt displays once for each unique certificate and for each node. After the signer for the node is added, the signer remains in the client truststore. The following sample code shows the signer exchange prompt retrieving a signer from a client:

```
/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/default/bin/serverStatus -all ADMU0116I:
Tool information is being logged in file
/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/default/logs/serverStatus.log ADMU0128I:
Starting tool with the default profile ADMU0503I: Retrieving server status for all servers
ADMU0505I: Servers found in
configuration: ADMU0506I: Server name: server1
*** SSL SIGNER EXCHANGE PROMPT *** SSL signer from target host 192.168.1.5
is not found in truststore
/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/default/etc/trust.p12.
Here is the signer information
(verify the digest value matches what is  displayed at the server):
Subject DN:    CN=myhost.austin.ibm.com, OU=myhostNode01Cell, OU=myhostNode01,
O=IBM, C=US Issuer DN:
CN=myhost.austin.ibm.com, O=IBM, C=US
Serial number: 2510775664686266 Expires:       Thu Feb 19 15:58:49 CST 2009
SHA-1 Digest:  2F:96:70:23:08:58:6F:66:CD:72:61:E3:46:8B:39:D4:AF:62:98:C3
MD5 Digest:    04:53:F8:20:A2:8A:6D:31:D0:1D:18:90:3D:58:B9:9D

Subject DN: CN=myhost.austin.ibm.com, OU=Root Certificate, OU=myhostNode01Cell,
OU=myhostNode01, O=IBM, C=US Issuer DN:  CN=myhost.austin.ibm.com, OU=Root
```

```
Certificate, OU=myhostNode01Cell, OU=myhostNode01, O=IBM, C=US Serial number:
2510773295548841 Expires: Tue Feb 15 15:58:46 CST 2028 SHA-1 Digest:
2F:96:70:23:08:58:6F:66:CD:72:61:E3:46:8B:39:D4:AF:62:98:C3
MD5 Digest: 04:53:F8:20:A2:8A:6D:31:D0:1D:18:90:3D:58:B9:9D

Add signer to the truststore now? (y/n) y A retry of the request may need to occur.
ADMU0508I: The Application Manager "server1" is STARTED
```

To automate this process, see “retrieveSigners command” on page 1730.

When a prompt occurs to accept the signer, a socket timeout can occur and the connection might be broken. For this reason, the message `A retry of the request may need to occur.` displays after answering the prompt. The message informs the user to resubmit the request. This problem should not happen frequently, and it might be more prevalent for some protocols than others.

A retry of the request may need to occur if the socket times out while waiting for a prompt response. If the retry is required, note that the prompt will not be re-displayed if `(y)` is entered, which indicates the signer has already been added to the trust store.

Verify the displayed SHA-1 digest, which is the signature of the certificate that is sent by the server. If you look at the certificate on the server, verify that the same SHA-1 digest displays.

You can disable the prompt when you do not want it to display by running the **retrieveSigners** utility to retrieve all of the signers for a particular cell. You can download or upload the signers from any remote keystore to any local keystore by referencing a common truststore with this client script. For more information, see “Default chained certificate configuration in SSL” on page 1724.

Using the retrieveSigners utility to download signers for a client

You can run the **retrieveSigners** utility to retrieve all of the signers from the remote keystore for a specified client keystore. The truststore contains the signers that enable the client to connect to its processes. The **retrieveSigners** utility can point to any keystore in the target configuration, within the scope of the target process, and can download the signers (certificate entries only) to any client keystore in the `ssl.client.props` file.

Obtaining signers for clients and servers from a previous release

Note: When a client from a release prior to version 7.0 connects to the current release, the client must obtain signers for a successful handshake. Clients using previous releases of WebSphere Application Server cannot obtain signers as easily as in the current release. You can copy the deployment manager *common* truststore to your back-level client or server, and then re-configure the SSL configuration to directly reference that truststore. This common truststore of type `PKCS12` is located in the `/config/cells/<cell_name>/nodes/<node_name>` directory in the configuration repository and has a default password of `WebAS`.

To collect all of the signers for the cell in a single `trust.p12` keystore file, complete following steps:

1. Copy the trust.p12 keystore file on the server and replicate it on the client. The client references the file directly from the `sas.client.props` and `soap.client.props` files that specify the SSL properties for previous releases.
2. Change the client-side keystore password so that it matches the default cell name that is associated with the copied keystore.
3. Change the default keystore type for the `trust.p12` file to `PKCS12` in the client configuration.

The following two code samples show you a before and an after view of the changes to make.

Default SSL configuration of sas.client.props for a previous release

```
com.ibm.ssl.protocol=SSL com.ibm.ssl.keyStore=/QIBM/UserData/WebSphere/AppServer/V8/Base/
profiles/default/
etc/DummyClientKeyFile.jks
com.ibm.ssl.keyStorePassword={xor}CDo9Hgw\= com.ibm.ssl.keyStoreType=JKS com.ibm.ssl.trustStore=
/QIBM/UserData/WebSphere/AppServer/
V8/Base/profiles/default/etc/DummyClientTrustFile.jks com.ibm.ssl.trustStorePassword={xor}CDo9Hgw\=
com.ibm.ssl.trustStoreType=JKS
```

SSL configuration changes that are required to common truststore file in the /etc directory of the client

```
com.ibm.ssl.protocol=SSL com.ibm.ssl.keyStore=/QIBM/UserData/WebSphere/AppServer/V8/Base/
profiles/default/etc/
DummyClientKeyFile.jks com.ibm.ssl.keyStorePassword={xor}CDo9Hgw\= com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.trustStore=/QIBM/UserData/WebSphere/AppServer/V8/Base/profiles/default/etc/trust.p12
com.ibm.ssl.trustStorePassword=myhostNode01Cell com.ibm.ssl.trustStoreType=PKCS12
```

Tip: You can run the **PropsFilePasswordEncoder** script, which is located in the `/bin` directory to encode the password.

You can also make these changes in the `soap.client.props` file and specify the `key.p12` file in place of the `DummyClientKeyFile.jks` file. However, you must also change the keyStorePassword and keyStoreType values to match those in the default `key.p12` file.

In releases of WebSphere Application Server prior to Version 7.0, you must edit the SSL configuration on the server to replace the common truststore. The `trust.p12` file, which is used by the server, also must contain the default dummy certificate signer for connections among servers at previous release levels. You might need to manually extract the default certificate from the `DummyServerKeyFile.jks` file and then import the certificate into the `trust.p12` file that you added to the configuration.

retrieveSigners command:

The retrieveSigners command creates a new client self-signed certificate, keystore, and SSL configuration in the ssl.client.props file. Using this command you can optionally extract the signer to a file.

For more information about where to run this command, read about Using command tools.

Syntax

Use the following command syntax to create a new client self-signed certificate, keystore, and SSL configuration in the ssl.client.props file.

```
retrieveSigners <remoteKeyStoreName> <localKeyStoreName> [options]
```

The <remoteKeyStoreName> and <localKeyStoreName> parameters are required. The following optional parameters are available:

```
[-remoteAlias aliasFromRemoteStore]
[-localAlias storeAsAlias]
[-listRemoteKeyStoreNames][-listLocalKeyStoreNames]
[-autoAcceptBootstrapSigner][-uploadSigners] [-host host]
[-port port][-conntype JSR160RMI|RMI|SOAP|IPC][-user user]
[-password password]
[-trace] [-logfile filename]
[-replacelog] [-quiet] [-help]
```

Parameters

The following parameters are available for the **retrieveSigners** command:

-remoteKeyStoreName
The name of a truststore that is located in the server configuration from which to retrieve the signers. This parameter is typically the CellDefaultTrustStore file for a managed environment or the NodeDefaultTrustStore file for an unmanaged environment.

-localKeyStoreName
The name of the truststore that is located in the ssl.client.props file for the profile to which the retrieved signers is added. This parameter is typically the ClientDefaultTrustStore file for either a managed or unmanaged environment.

-remoteAlias <aliasFromRemoteStore>
Specifies one alias from the remote truststore that you want to retrieve. Otherwise, all signers from the remote truststore are retrieved.

-localAlias <storeAsAlias>
Determines the name of the alias stored in the local truststore. This option is only valid if you specify the –remoteAlias option. If you do not specify the -localAlias option, then the alias name from the remote truststore is used, if possible. If an alias clash occurs, then the alias name is used and has an incremented number appended to the end of it until a unique alias is found.

-listRemoteKeyStoreNames
Sends a remote request to the server to list all keystores that you can specify for the remoteKeyStoreName parameter. Use this command when you are unsure of the name of the remote truststore from which you want to download the signers.

-listLocalKeyStoreNames
Lists the keystores located in the ssl.client.props file that you can specify for the localKeyStoreName parameter. This truststore receives the signers from the server. Use this parameter when you are unsure of the name of the local truststore into which you want to retrieve the signers. The default name of the truststore is ClientDefaultTrustStore and is located in the ssl.client.props file.

-autoAcceptBootstrapSigner
Automatically adds a signer to make a secure connection to the server. The purpose of the option is to support automation of the command so that you do not need to accept the signer. After the signer is added to the local truststore, an SHA hash prints so that you can verify the certificate.

-uploadSigners
Converts the signer download into a signer upload. The signers from the localKeyStoreName parameter is sent to the remoteKeyStoreName parameter instead.

-host <host>
Specifies the target host from which the signers are retrieved.

-port <port>
Specifies the target administrative port to which you want to connect. You must specify the port based on the -conntype parameter. If the conntype is SOAP, the default port is 8879. This value can vary for different servers. If the conntype is RMI, the default port is 2809.

-conntype <JSR160RMI|IPC|RMI|Soap>
Determines the administrative connector type that is used for the MBean call to retrieve the signers.

Note: Eventually switch from the RMI connector to the JSR160RMI connector because support for the RMI connector is deprecated.

-user <user>
When the -uploadSigners flag is used, you are required to specify this option to supply the user name that is authenticated for the MBean operation. If you do not specify this parameter when the -uploadSigners flag is used, then you are prompted for credentials by default.

-password <password>
When the -uploadSigners flag is used, you are required to specify this option to supply the password that is authenticated for the MBean operation. The password goes along with the –user parameter.

-trace
When specified, this parameter enables tracing of the trace specification necessary to debug this component. By default, the trace is located in the profiles/*profile_name*/log/retrieveSigners.log file.

-logfile <filename>
Overrides the default trace file. By default, the trace will appear in the profiles/*profile_name*/log/retrieveSigners.log file.

-replacelog
Causes the existing trace file to be replaced when the command runs.

-quiet
Suppresses most messages from printing to the console.

-help
Prints a usage statement.

-? Prints a usage statement.

Usage scenario

The following examples demonstrate correct syntax for using the **retrieveSigners** command:

- The following example lists remote and local keystores:

```
retrieveSigners -listRemoteKeyStoreNames -listLocalKeyStoreNames -conntype RMI -port 2809
```

Example output

```
CWPKI0306I: The following remote keystores exist on the specified server:
          CMSKeyStore, NodeLTPAKeys, NodeDefaultTrustStore, NodeDefaultKeyStore
CWPKI0307I: The following local keystores exist on the client:
          ClientDefaultKeyStore, ClientDefaultTrustStore
```

- The following example retrieves all signers from NodeDefaultTrustStore:

```
retrieveSigners NodeDefaultTrustStore ClientDefaultTrustStore -autoAcceptBootstrapSigner
-conntype RMI -port 2809
```

Example output

```
CWPKI0308I: Adding signer alias "CN=BIRKT40.austin.ibm.com, O=IBM, C=US" to
          local keystore "ClientDefaultTrustStore" with the following SHA
          digest: 40:20:CF:BE:B4:B2:9C:F0:96:4D:EE:E5:14:92:9E:37:8D:51:A5:47
```

Certificate expiration monitoring in SSL:

The certificate expiration monitor administrative task is a scheduled task that cycles through all the keystores in the security configuration and reports on any certificates that are expired, certificates that fall within the expiration threshold, and certificates that fall within the pre-notification period.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

The certificate expiration monitor also replaces self-signed and chained certificates that have a root in the root keystore if configured to do so. If the monitor is configured to replace certificates, then certificates that are expired or fall in the expiration threshold are replaced. Certificates that are imported from an external Certificate authority (CA) are reported but not replaced.

Certificate expiration monitoring relies on the following definitions:

Expired certificates
Certificates are created with a finite life span. Self-signed or chained certificates that have reached the end of their life span are reported and replaced, if possible. Certificate authority signed certificates cannot be replaced but will be reported. Replacing CA-signed certificates is the responsibility of the administrator.

Certificates within the expiration threshold

There is a period of time before a certificate expires. A certificate in this period of time is one within the expiration threshold. The server replaces certificates within the expiration threshold so that the certificate does not expire and cause outages. By default the expiration threshold is 60 days, but can be configured as required.

Pre-notification period

Before a certificate falls within the expiration threshold there are warnings issued that indicate that the certificate will be replaced, when the expiration threshold date is reached. The period of time prior to the expiration threshold date is called the pre-notification period and is set at 90 days for the certificate.

The certificate expiration monitor performs the following:

1. Clears out the **NodeDefaultDeletedStore** or **DmgrDefaultDeletedStore**. This operation is performed silently without reporting that the certificates are deleted.
2. Checks the root key stores, **DmgrDefaultRootStore** or **NodeDefaultRootStore** and the **DmgrRSATokenRootStore** or **NodeRSATokenRootStore**. If any root certificates are expired, falls in the threshold period, or the pre-notification period, then the certificate is noted in the report.
3. If there are any root certificates that are expired or fall in the threshold period that root certificate is recreated using all the information used to create the original one. Any signer certificates from the original root certificate are replaced with the signers from the new root certificate.
4. If a root certificate is replaced, then all the keystores are checked to see if there are any chained certificates signed with the original root certificate. If there are, then the chain certificate is renewed (recreated with the new root certificate). Any signer certificate from the original certificate is replaced with the signer from the recreated certificate.
5. After all root keystores are processed, the rest of the keystores are checked for expired certificates, certificates in the expiration threshold, or certificates in the pre-notification period. Any certificate falling in any one of these categories is noted in the report.
6. If there are any expired certificates or certificates in the expiration threshold period and these certificates are self-signed certificates or chained certificates created by WebSphere, then they are replaced. If the chained certificates root is not in the root key store then it will be recreated as a default root certificate. Any signer certificates from the original certificate are replaced with the signer from the new certificate."
7. A report is generated and returned, written to a log file, or mailed.

The server default certificate is a chained certificate with a 365 day life span. It is signed with the default root certificate which has a 15 year life span.

You can configure this monitor task to run according to a particular schedule. The schedule produces the next start date that persists in the configuration and, when the date is reached, WebSphere Application Server starts the monitor to check all of the keystores for certificates that meet the expiration threshold. You can start the task manually to run at any time.

The following `security.xml` configuration object specifies when the monitor task starts, determines the certificate expiration threshold, and indicates whether you are notified in an email using Simple Mail Transfer Protocol (SMTP) or in a message log.

```
<wsCertificateExpirationMonitor xmi:id="WSCertificateExpirationMonitor_1"
name="Certificate Expiration Monitor" daysBeforeNotification="30"
isEnabled="true" autoReplace="true" deleteOld="true"
wsNotification="WSNotification_1" wsSchedule="WSSchedule_2"
nextStartDate="1134358204849"/>
```

The expiration monitor replaces self-signed certificates and chained personal certificates that are signed by a root certificate in **DmgrDefaultRootStore** or **NodeDefaultRootStore**. Self-signed certificates are renewed using all the information that was used to create the original self-signed certificate. A chained certificate is renewed using the same root certificate that was used to sign the original certificate.

The expiration monitor automatically replaces only self-signed certificates and chained certificates that are expired or that meet the expiration threshold criteria. To replace all of the signers from the old certificate with the signer that belongs to the new certificate in all the keystores in the configuration for that cell, set the autoReplace attribute to `true`. When the deleteOld attribute is `true`, the old personal certificate and old signers also are deleted from the keystores. The isEnabled attribute determines whether the expiration monitor task runs based upon the nextStartDate attribute that is derived from the schedule. The nextStartDate attribute is derived from the schedule in milliseconds since 1970, and is identical to the System.currentTimeMillis(). If the nextStartDate has already passed when an expiration monitor process begins, and the expiration monitor is enabled, the task is started, but a new nextStartDate value is established based on the schedule.

The following sample the schedule object shows the frequency attribute as the number of days between each run of the certificate monitor.

```
<wsSchedules xmi:id="WSSchedule_2" name="ExpirationMonitorSchedule"
frequency="30" dayOfWeek="1" hour="21" minute="30"/>
```

The dayOfWeek attribute adjusts the schedule to run on a specified day of the week, which is always the same day regardless of whether the frequency is set to 30 or 31 days. Based on 24-hour clock, the hour and minute attributes determine when the expiration monitor is started on the specified day.

The following sample code of the notification object shows the notification configuration, which notifies you after the expiration monitor runs.

```
<>wsNotifications xmi:id="WSNotification_1" name="MessageLog" logToSystemOut="true" emailList=""/
```

For expiration monitor notifications, you can select message log, email using SMTP server, or both methods of notification. When you configure the email option, use the format `user@domain@smtpserver`. If you do not specify an SMTP server, WebSphere Application Server defaults to the same domain as the email address. For example, if you configure joeuser@ibm.com, WebSphere Application Server attempts to call smtp-server.ibm.com. To specify multiple email addresses using scripting, you must add a pipe (I) character between entries. When you specify the logToSystemOut attribute, the expiration monitor results are sent to the message log for the environment, which is typically the `SystemOut.log` file.

The expiration monitor clears out the deleted certificates keystore. The monitor first clears out the deleted keystore. Due to the nature of the PKCS12 keystore, there must be at lease one entry in the keystore so the signer certificates from the dummy key store will remain in the deleted keystore. There is no reporting on the certificate being deleted from the deleted keystore.

Important: When the expiration monitor replaces certificates, this can dynamically affect the runtime when the following configuration option is enabled:

Security > **SSL certificate and key management**. Under configuration settings, check the checkbox for **Dynamically update the run time when SSL configuration changes occur**.

When enabled, any certificates that are replaced causes the client SSL runtime to begin using the new certificates immediately, which in turn, flushes SSL and keystore caches and causes some ports using SSLServerSockets (RMI/IIOP on distributed and Admin SOAP) to restart. Restarting ports breaks existing connections. These connections can be reconnected after the port restart is completed. Endpoints using the channel framework (HTTP, BUS, RMI/IIOP on z/OS) leave existing connections unaffected but still use the new certificates for new connections.

When the dynamic change property is disabled and before the new certificates become effective, the administrator needs to recycle all processes in the entire cell after each node has the synchronized configuration. Regardless of which method is chosen, you should always check the health of your cell after the certificate expiration monitor has run (based on the

schedule specified). The schedule should be set to run the certificate expiration monitor during a maintenance period so that if a restart is required after the certificate replacement, it will not cause unexpected outages.

Dynamic configuration updates in SSL

During the Secure Sockets Layer (SSL) runtime, dynamic configuration updates affect both inbound and outbound SSL endpoints. For inbound SSL endpoints, the changes that are implemented by the SSL channel are only affected by dynamic changes. For outbound SSL endpoints, all outbound connections inherit the new configuration changes.

In this release, dynamic update functionality provides you with greater flexibility and efficiency. You can change SSL configurations without restarting WebSphere Application Server for the changes to take effect.

To make dynamic changes, in the administrative console click **Security > SSL certificates and key management**, then select the **Dynamically update the runtime when SSL configuration changes occur** check box. You must save your changes and then synchronize the `security.xml` file with remote systems. A remote system must be able to confirm that `dynamicallyUpdateSSLConfig=true` is in the `security.xml` file.

The SSL runtime reloads the modified SSL configuration and creates a new SSLEngine for the modified connections that are associated with inbound endpoints. New outbound connections use the new configuration while existing connections continue to use the old SSLEngine object and are not affected.

Tip: Make dynamic changes to the SSL configuration during off-peak hours. Synchronization delays can negatively affect connections when you update SSL configurations during peak hours.

You can turn on and off the dynamicallyUpdateSSLConfig attribute in the `security.xml` file to ensure successful updates by doing the following actions:

1. Set dynamicallyUpdateSSLConfig=On.
2. Save the updated configuration.
3. Synchronize the `security.xml` file with remote systems.
4. Set the dynamicallyUpdateSSLConfig attribute to `Off`.

You must verify that all of the nodes receive the changes before turning off the dynamicallyUpdateSSLConfig attribute. Test the changes in a test environment before updating the production environment.

Tip: Some SSL changes, especially administrative SSL changes, can cause server outages if you fail to test them first. When a change prevents trust between two endpoints, the endpoints cannot communicate with each other. Additionally, if administrative SSL connection updates cause system outages, you might need to disable the nodes after you make corrective changes using the deployment manager. From the command line, you can manually synchronize the server to retrieve the new SSL changes, then restart the nodes.

Certificate management using iKeyman prior to SSL

Starting in WebSphere Application Server Version 6.1, you can manage your certificates from the administrative console. When using versions of WebSphere Application Server prior to Version 6.1, use iKeyman for certificate management. iKeyman is a key management utility.

WebSphere Application Server certificate management requires that you define the keystores in your WebSphere Application Server configuration. With iKeyman, you need access to the keystore file only. You can read a keystore file with personal certificates and signers that is created in iKeyman. A keystore file can be read into the WebSphere Application Server configuration by using the **createKeyStore** command.

The majority of certificate management functions are the same between WebSphere Application Server and iKeyman, especially for personal certificates and signer certificates. However, certificate requests are

special. The underlying behavior is different in the two certificate management schemes. Because of the different behavior, when a certificate request is generated from iKeyman, the process must be completed in iKeyman. For example, a certificate that is generated by a certificate request that originated in iKeyman must be received in iKeyman as well.

The same is true for WebSphere Application Server. For example, when a certificate is generated from a certificate request that originated in WebSphere Application Server, the certificate must be received in WebSphere Application Server.

You can perform the following certificate operations using iKeyman:

Table 142. Available certificate operations using iKeyman. This table describes the available certificate operations using iKeyman.

Types of certificates	Functions	Description
Personal certificates	Create a self-signed certificate	Creates a self-signed certificate and stores it in a keystore.
	List personal certificates	Lists all the personal certificates in a keystore.
	Get information about a personal certificate	Gets information about a personal certificate.
	Delete a personal certificate	Deletes a personal certificate from a keystore.
	Import a certificate	Imports a certificate from a keystore to a keystore.
	Export a certificate	Exports a certificate from a keystore to another keystore.
	Extract a certificate	Extracts the signer part of a personal certificate to a file.
	Receive a certificate	Reads a certificate that comes from a certificate authority (CA) into a keystore.
Signer certificates	Add a signer certificate	Adds a signer certificate from a file to a keystore.
	List signer certificates	Lists all the signer certificates in a keystore.
	Get information about a signer certificate	Gets information about a signer certificate.
	Delete a signer certificate	Deletes a signer certificate from a keystore.
	Extract a signer certificate	Extracts a signer certificate from a keystore, and stores the certificate in a file.
Certificate requests	Create a certificate request	Creates a certificate request that can be sent to a CA.
	List certificate requests	Lists the certificate requests in a keystore.
	Get information about a certificate request	Gets information about a certificate request.
	Delete a certificate request	Deletes a certificate request from a keystore.
	Extract a certificate request	Extracts a certificate request to a file.

Certificate management in SSL

You can manage certificate operations that involve personal certificates, signer certificates, and personal certificate requests on the administrative console.

Types of certificates

WebSphere Application Server uses the certificates that reside in keystores to establish trust for a Secure Sockets Layer (SSL) connection. Click **Security > SSL certificate and key management > Manage**

endpoint security configurations > Inbound | Outbound > *SSL_configuration_name* > Key stores and certificates, then select an existing or create a new keystore. After selecting a keystore, and depending on the type of certificate you need, choose one of the following types of certificates under Related Items:

- Personal certificate
- Signer certificate
- Certificate Authority (CA) certificates
- Personal certificate request

Table 143. Certificate operations. The following table describes the certificate operations that you can perform on the administrative console

Types of certificates	Functions	Description
Personal certificates	Create a self-signed certificate	Creates a self-signed certificate and stores it in a keystore.
	List personal certificates	Lists all the personal certificates in a keystore.
	Get information about a personal certificate	Gets information about a personal certificate.
	Delete a personal certificate	Deletes a personal certificate from a keystore.
	Import a certificate	Imports a certificate from a keystore to a keystore.
	Export a certificate	Exports a certificate from a keystore to another keystore.
	Extract a certificate	Extracts the signer part of a personal certificate to a file.
	Exchange signer certificates	Exchange signer part of a personal certificate between key store.
	Receive a certificate	Reads a certificate that comes from a certificate authority (CA) into a keystore.
	Replace a certificate	Replaces all occurrences of a personal certificate alias in the WebSphere Application Server configuration with another certificate. Also, replaces all occurrences of the personal certificates signer with the new personal certificate signer.
	Create a chained certificate	Creates a chained certificate and stores it in a keystore.
	Renew a certificate	Renews a certificate with a new public/private key pair and stores it in a keystore.
	Request a CA certificate	Makes a request to a CA using a CA client to obtain a CA certificate.
Certificate authority (CA) certificates	Create CA certificate	Sends a certificate request to an external certificate authority (CA).
	Revoke CA certificate	Sends a revocation request to an external certificate authority (CA).
Signer certificates	Add a signer certificate	Adds a signer certificate from a file to a keystore.
	List signer certificates	Lists all the signer certificates in a keystore.
	Get information about a signer certificate	Gets information about a signer certificate.
	Delete a signer certificate	Deletes a signer certificate from a keystore.
	Extract a signer certificate	Extracts a signer certificate from a keystore, and stores the certificate in a file.

Table 143. Certificate operations (continued). The following table describes the certificate operations that you can perform on the administrative console

Types of certificates	Functions	Description
	Retrieve a signer from a port	Retrieves a signer certificate from a port, and stores it in a key store.
Certificate requests	Create a certificate request	Creates a certificate request that can be sent to a CA.
	List certificate requests	Lists the certificate requests in a keystore.
	Get information about a certificate request	Gets information about a certificate request.
	Delete a certificate request	Deletes a certificate request from a keystore.
	Extract a certificate request	Extracts a certificate request to a file.

Personal certificates

Table 144. Personal certificate operations. The following table lists the operations that you can perform on personal certificates, the AdminTask object that you can use to perform that operation, and how to navigate to the certificate on the console:

Function	AdminTask object	Administrative console
Create a self-signed certificate	createSelfSignedCertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> Create a Self-Signed Certificate**
List personal certificates	listPersonalCertificates	**Security > Secure Communications > Key store and certificates >** *key store* **> personal certificates**
Get information about a personal certificate	getPersonalCertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> personal certificates >** *alias*
Delete a personal certificate	deletePersonalCertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> personal certificates > delete**
Import a certificate	importCertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> personal certificates > import**
Export a certificate	exportCertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> personal certificates > export**
Extract a certificate	extractCertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> personal certificates > extract**
Exchange signer certificates	exchangeSignerCertificates	**Security > Secure Communications > Key store and certificates > Exchange signers**
Create a chained certificate	createChainedCertificate	**Security > SSL certificate and key management > Key store and certificates >** ***keystore name*** **> Personal certificates**. Click **Create** button and select **Chained certificate**
Renew a certificate	renewChainedCertificate	**Security > SSL certificate and key management > Key store and certificates >** ***keystore name*** **> Personal certificates**. Select a certificate. Click **Renew** button.
Create a chained Certificate	createChainedCertificate	**Security > Secure communications > Key store and certificates > keystore > Create a chained certificate**.

Table 144. Personal certificate operations (continued). The following table lists the operations that you can perform on personal certificates, the AdminTask object that you can use to perform that operation, and how to navigate to the certificate on the console:

Function	AdminTask object	Administrative console
Request a CA certificate	requestCACertificate	**Security > Secure communications > Key store and certificates > keystore > Request a CA certificate**.

Certificate authority (CA) certificates

Table 145. CA certificate operations. The following table lists the operations that you can perform on CA certificates, the AdminTask object that you can use to perform that operation, and how to navigate to the certificate on the console:

Function	AdminTask object	Administrative console
Create a CA certificate	createCACertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> Personal certificates > Create > CA-signed certificate**
Revoke a CA certificate	revokeCACertificate	**Security > Secure Communications > Key store and certificates >** *key store* **> Personal certificates** *personal certificate* **> Revoke**

Signer certificates

Table 146. Signer certificate operations. The following table lists the operations that you can perform with signer certificates, the AdminTask object that you can use to perform the operation, and how to navigate to the certificate on the console:

Function	AdminTask object	Administrative console
Add a signer certificate	addSignerCertificate	**Security > Secure communications > Key store and certificates >** *key store* **> signer certificates > Add**
List signer certificates	listSignerCertificates	**Security > Secure communications > Key store and certificates >** *key store* **> signer certificates**
Get information about a signer certificate	getSignerCertificate	**Security > Secure communications > Key store and certificates >** *key store* **> signer certificates >** *alias*
Delete a signer certificate	deleteSignerCertificate	**Security > Secure communications > Key store and certificates >** *key store* **> signer certificate >delete**
Extract a signer certificate to a file	extractSignerCertificate	**Security > Secure communications > Key store and certificates >** *key store* **> signer certificates > extract**
Retrieve a signer certificate from a port	retrieveSignerFromPort	**Security > Secure communications > Key store and certificates >** *key store* **> signer certificates > retrieve from port**

Personal certificate requests

Table 147. Personal certificate request operations. The following table lists the operations that you can perform on personal certificate requests, the AdminTask object that you can use to perform that operation, and how to navigate to the certificate request on the console:

Function	AdminTask object	Administrative console
Create a personal certificate request	createCertificateRequest	**Security > Secure communications > Key store and certificates >** *key store* **> Personal certificate Requests > Add**

Table 147. Personal certificate request operations (continued). The following table lists the operations that you can perform on personal certificate requests, the AdminTask object that you can use to perform that operation, and how to navigate to the certificate request on the console:

Function	AdminTask object	Administrative console
List personal certificate requests	listCertificateRequests	**Security > Secure communications > Key store and certificates >** *key store* **> Personal certificate requests**
Get information about a personal certificate request	getCertificateRequest	**Security > Secure communications > Key store and certificates >** *key store* **> Personal certificate requests >** *alias*
Delete a personal certificate request	deleteCertificateRequest	**Security > Secure communications > Key store and certificates >** *key store* **> Personal certificate requests > delete**
Extract a personal certificate request to a file	extractCertificateRequest	**Security > Secure communications > Key store and certificates >** *key store* **> Personal certificate requests > Extract**

Using the retrieveSigners command in SSL to enable server to server trust

You can add a signer certificate to a server's `trust.p12` file, allowing that server to securely communicate with another server. This can be done using the **retrieveSigners** command to add a signer to a server's `trust.p12` file after making changes to the `ssl.client.props` file.

Before you begin

The server that will be communicating as a client must be identified before the server to server trust can be established. You will make change to the `ssl.client.props` file and run the **retrieveSigners** command on the server communicating as a client. If both servers will be acting as a client , these steps will be required for both servers.

About this task

The `ssl.client.props` file is setup by default to configure Secure Socket Layer (SSL) communication for clients. This makes the default behavior of the **retrieveSigners** command work on the client's `trust.p12` file and `key.p12` file in the *profile_root*`/etc` directory. You can add a signer certificate to a server's `trust.p12` file, allowing that server to act as a client communicating to another server. Using the **retrieveSigners** command to add a signer to a server's `trust.p12` file requires some changes to the `ssl.client.props` file.

Procedure

1. Open the `ssl.client.props` file. The `ssl.client.props` file is located in *profile_root*`/properties` ditrectory.
2. Uncomment the section of `ssl.client.props` that starts with com.ibm.ssl.alias=AnotherSSLSettings property.
3. Uncomment the section of `ssl.client.props` that starts with com.ibm.ssl.trustStoreName=AnotherTrustStore property.
4. Enter the location of the trust store that the signer should be added. If you are using the server trust store for a deployment manager then it is located in *profile_root*`/config/cells/`*cell name*`/trust.p12`. If using the trust store for an application server, it is located in *profile_root*`/config/cells/`*cell name*`/nodes/`*node name*`/trust.p12`.
5. Update the remaining properties in this section with the values associated with the trust store being used. A description of the properties can be found in ssl.client.props client configuration file.
6. Optional: Uncomment and update section that starts with com.ibm.ssl.trustStoreName=AnotherKeyStore property. Most scenarios only require a signer to be

added to the trust store. This example only adds a signer to the trust store, but you can also add a signer to the key store by updating the properties as you did for the trust store in steps 3 through 5.

7. Save the changes made to `ssl.client.props`.
8. Run the **retrieveSigners** command. For more information see the page about the retrieveSigners command.

```
retrieveSigners NodeDefaultTrustStore AnotherTrustStore -host ademyers.austin.ibm.com -port 8879
```

Example output:

```
CWPKI0308I: Adding signer alias "default_1" to local keystore
                    "AnotherTrustStore" with the following SHA digest:
                    F4:71:97:79:3E:C1:DC:E7:9F:8F:3D:F0:A0:15:1E:D1:44:73:2C:06
```

Results

After the steps have been successfully completed. the server acting as a client has the signing certificate of the other server. This allows that server to establish a SSL connection to the other server.

Example

The example shows the modified section of the `ssl.client.props` file assuming that the server's `trust.p12` file is being used. Any trust store existing trust store can be used if the properties are provided for that trust store.

```
#-------------------------------------------------------------------------
com.ibm.ssl.alias=AnotherSSLSettings
com.ibm.ssl.protocol=SSL_TLS
com.ibm.ssl.securityLevel=HIGH
com.ibm.ssl.trustManager=IbmX509
com.ibm.ssl.keyManager=IbmX509
com.ibm.ssl.contextProvider=IBMJSSE2
com.ibm.ssl.enableSignerExchangePrompt=true
#com.ibm.ssl.keyStoreClientAlias=default
#com.ibm.ssl.customTrustManagers=
#com.ibm.ssl.customKeyManager=
#com.ibm.ssl.dynamicSelectionInfo=
#com.ibm.ssl.enabledCipherSuites=

# KeyStore information
#com.ibm.ssl.keyStoreName=AnotherKeyStore
#com.ibm.ssl.keyStore=${user.root}/etc/key.p12
#com.ibm.ssl.keyStorePassword={xor}CDo9Hgw=
#com.ibm.ssl.keyStoreType=PKCS12
#com.ibm.ssl.keyStoreProvider=IBMJCE
#com.ibm.ssl.keyStoreFileBased=true

# TrustStore information
com.ibm.ssl.trustStoreName=AnotherTrustStore
com.ibm.ssl.trustStore=${user.root}/config/cells/localhostCell01/trust.p12
com.ibm.ssl.trustStorePassword={xor}CDo9Hgw=
com.ibm.ssl.trustStoreType=PKCS12
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
```

What to do next

After the signer has been added, edit the `ssl.client.props` file to comment out the sections that were to used to add the signer certificate.

Creating a Secure Sockets Layer configuration

Secure Sockets Layer (SSL) configurations contain the attributes that you need to control the behavior of client and server SSL endpoints. You create SSL configurations with unique names within specific management scopes on the inbound and outbound tree in the configuration topology. This task shows you how to define SSL configurations, including quality of protection and trust and key manager settings.

Before you begin

You must decide at which scope you need to define an SSL configuration, for instance, the cell, node group, node, server, cluster, or endpoint scope, from the least specific to the most specific scope. When

you define an SSL configuration at the node scope, for example, only those processes within that node can load the SSL configuration; however, any processes at the endpoint in the cell can use an SSL configuration at the cell scope, which is higher in the topology.

You must also decide which scope to associate with the new SSL configuration, according to the processes that the configuration affects. For example, an SSL configuration for a hardware cryptographic device might require a keystore that is available only on a specific node, or you might need an SSL configuration for a connection to a particular SSL host and port. For more information, see “Dynamic outbound selection of Secure Sockets Layer configurations” on page 1717.

gotcha: The security.xml file is restricted. Therefore, if you need to make changes to the security.xml file, verify that your user ID has administrator role authorization. If you are using a user ID with operator role authorization, you can perform a node synchronization, but any changes that you made to the security.xml file are not synchronized.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations**.
2. Select an SSL configuration link on either the Inbound or Outbound tree, depending on the process you are configuring.
   - If the scope is already associated with a configuration and alias, the SSL configuration alias and certificate alias are noted in parentheses.
   - If the parenthetical information is not included, then the scope is not associated. Instead, the scope inherits the configuration properties of the first scope above it that is associated with an SSL configuration and certificate alias.

   The cell scope must be associated with an SSL configuration because it is at the top of the topology and represents the default SSL configuration for the inbound or outbound connection.
3. Click **SSL configurations** under Related Items. You can view and select any of the SSL configurations that are configured at this scope. You can also view and select these configuration at every scope that is lower on the topology.
4. Click **New** to display the SSL configuration panel. You cannot select links under Additional Properties until you type a configuration name and click **Apply**.
5. Type an SSL configuration name. This field is required. The configuration name is the SSL configuration alias. Make the alias name unique within the list of SSL configuration aliases that are already created at the selected scope. The new SSL configuration uses this alias for other configuration tasks.
6. Select a truststore name from the drop-down list. A truststore name refers to a specific truststore that holds signer certificates that validate the trust of certificates sent by remote connections during an SSL handshake. If there is no truststore in the list, see “Creating a keystore configuration for a preexisting keystore file” on page 1792 to create a new truststore, which is a keystore whose role is to establish trust during the connection.
7. Select a keystore name from the drop-down list. A keystore contains the personal certificates that represent a signer identity and the private key that WebSphere Application Server uses to encrypt and sign data.
   - If you change the keystore name, click **Get certificate aliases** to refresh the list of certificates from which you can choose a default alias. WebSphere Application Server uses a server alias for inbound connections and a client alias for outbound connections.
   - If there is no keystore in the list, see “Creating a keystore configuration for a preexisting keystore file” on page 1792 to create a new keystore.

8. Choose a default server certificate alias for inbound connections. Select the default only when you have not specified an SSL configuration alias elsewhere and have not selected a certificate alias. A centrally managed SSL configuration tree can override the default alias. For more information, see “Central management of SSL configurations” on page 1718.
9. Choose a default client certificate alias for outbound connections. Select the default only when the server SSL configuration specifies an SSL client authentication.
10. Review the identified management scope for the SSL configuration. Make the management scope in this field identical to the link you selected in Step 2. If you want to change the scope, you must click a different link in the topology tree and continue at Step 3.
11. Click **Apply** if you intend to configure Additional Properties. If not, go to Step 24.
12. Click **Quality of protection (QoP) settings** under Additional Properties. QoP settings define the strength of the SSL encryption, the integrity of the signer, and the authenticity of the certificate.
13. Select a client authentication setting to establish an SSL configuration for inbound connections and for clients to send their certificates, if appropriate.
    - If you select **None**, the server does not request that a client send a certificate during the handshake.
    - If you select **Supported**, the server requests that a client send a certificate. However, if the client does not have a certificate, the handshake might still succeed.
    - If you select **Required**, the server requests that a client send a certificate. However, if the client does not have a certificate, the handshake fails.

    Important: The signer certificate that represents the client must be in the truststore that you select for the SSL configuration. By default, servers within the same cell trust each other because they use the common truststore, `trust.p12`, that is located in the cell directory of the configuration repository. However, if you use keystores and truststores that you create, perform a signer exchange before you select either **Supported** or **Required**.
14. Select a protocol for the SSL handshake.
    - The default protocol, SSL_TLS, supports client protocols TLSv1, SSLv3, and SSLv2.
    - The TLSv1 protocol supports TLS and TLSv1. The SSL server connection must support this protocol for the handshake to proceed.
    - The SSLv3 protocol supports SSL and SSLv3. The SSL server connection must support this protocol for the handshake to proceed.

    Important: Do not use the SSLv2 protocol for the SSL server connection. Use it only when necessary on the client side.
15. Select one of the following options:
    - A predefined Java Secure Socket Extension (JSSE) provider. The IBMJSSE2 provider is recommended for use on all platforms which support it. It is required for use by the channel framework SSL channel. When Federal Information Processing Standard (FIPS) is enabled, IBMJSSE2 is used in combination with the IBMJCEFIPS crypto provider.
    - A custom JSSE provider. Type a provider name in the **Custom provider** field.
16. Select from among the following cipher suite groups:
    - **Strong**: WebSphere Application Server can perform 128-bit confidentiality algorithms for encryption and support integrity signing algorithms. However, a strong cipher suite can affect the performance of the connection.
    - **Medium**: WebSphere Application Server can perform 40-bit encryption algorithms for encryption and support integrity signing algorithms.
    - **Weak**: WebSphere Application Server can support integrity signing algorithms but not to perform encryption. Select this option with care because passwords and other sensitive information that cross the network are visible to an Internet Protocol (IP) sniffer.

- **Custom**: you can select specific ciphers. Any time you change the ciphers that are listed from a specific cipher suite group, the group name changes to Custom.

17. Click **Update selected ciphers** to view a list of the available ciphers for each cipher strength.
18. Click **OK** to return to the new SSL configuration panel.
19. Click **Trust and key managers** under Additional Properties.
20. Select a default trust manager for the primary SSL handshake trust decision.
    - Choose **IbmPKIX** when you require certificate revocation list (CRL) checking using CRL distribution points in the certificates or the online certificate status protocol (OCSP).
    - Choose **IbmX509** when you do not require CRL checking but do need increased performance. You can configure a custom trust manager to perform CRL checking, if necessary.
21. Define a custom trust manager, if appropriate. You can define a custom trust manager that runs with the default trust manager you select. The custom trust manager must implement the JSSE javax.net.ssl.X509TrustManager interface and, optionally, the com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface to obtain product-specific information.
    a. Click **Security > SSL certificate and key management > Manage endpoint security configurations > *SSL_configuration* > Trust and key managers > Trust managers > New**.
    b. Type a unique trust manager name.
    c. Select the **Custom** option.
    d. Type a class name.
    e. Click **OK**. When you return to the Trust and key managers panel, the new custom trust manager displays in the **Additional ordered trust managers** field. Use the left and right list boxes to add and remove custom trust managers.
22. Select a key manager for the SSL configuration. By default, IbmX509 is the only key manager unless you create a custom key manager.

    Important: If you choose to implement your own key manager, you can affect the alias selection behavior because the key manager is responsible for selecting the certificate alias from the keystore. The custom key manager might not interpret the SSL configuration as the WebSphere Application Server key manager IbmX509 does. To define a custom key manager, click **Security > Secure communications > SSL configurations > *SSL_configuration* > Trust and key managers > Key managers > New**.
23. Click **OK** to save the trust and key manager settings and return to the new SSL configuration panel.
24. Click **Save** to save the new SSL configuration.

Results

Important: You can override the default trust manager when you configure at least one custom trust manager and set the com.ibm.ssl.skipDefaultTrustManagerWhenCustomDefined property to `true`. Click **Custom Property** on the SSL configuration panel. However, if you change the default, you leave all the trust decisions to the custom trust manager, which is not recommended for production environments. In test environments, use a dummy trust manager to avoid certificate validation. Remember that these environment are not secure.

What to do next

In this release of WebSphere Application Server, you can associate SSL configurations with protocols using one of the following methods:

- Set the SSL configuration on the thread programmatically
- Associate the SSL configuration with an outbound protocol, and target host and port. For more information, see “Associating a Secure Sockets Layer configuration dynamically with an outbound protocol and remote secure endpoint” on page 1771

- Associate the SSL configuration directly using the alias. For more information, see “Selecting an SSL configuration alias directly from an endpoint configuration” on page 1775
- Manage the SSL configurations centrally by associating them with SSL configuration groups or zones that are scoped for endpoints. For more information, see “Associating Secure Sockets Layer configurations centrally with inbound and outbound scopes” on page 1774.

SSL certificate and key management

Use this page to configure security for Secure Socket Layer (SSL) and key management, certificates, and notifications. The SSL protocol provides secure communications between remote server processes or endpoints. SSL security can be used for establishing communications inbound to and outbound from an endpoint. To establish secure communications, a certificate and an SSL configuration must be specified for the endpoint.

To view this administrative console page, click **Security > SSL certificate and key management**.

Configuration settings:

Specifies the following administrative console tasks:

- Manage endpoint security configurations
- Manage certificate expiration

Use Federal Information Processing Standard (FIPS) algorithms:

Specifies the Federal Information Processing Standard (FIPS)-compliant Java cryptography engine is enabled.

- Does not affect the SSL cryptography that is performed by the application server for z/OS System Secure Sockets Layer (SSSL).
- Does not change the JSSE provider if this cell includes any Application Server versions before the application server for z/OS Version 6.0.x.

When you select the **Use the Federal Information Processing Standard (FIPS)** option, the Lightweight Third Party Authentication (LTPA) implementation uses IBMJCEFIPS. IBMJCEFIPS supports the Federal Information Processing Standard (FIPS)-approved cryptographic algorithms for Data Encryption Standard (DES), Triple DES, and Advanced Encryption Standard (AES). Although the LTPA keys are backwards compatible with prior releases of the application server, the LTPA token is not compatible with prior releases. In prior releases, the application server did not generate the LTPA token using a FIPS-approved algorithm.

The IBMJSSE2 JSSE provider does not perform cryptographic functions directly, and therefore does not need to be FIPS-approved. Instead, the IBMJSSE2 JSSE provider uses the JCE framework for cryptographic functions and uses IBMJCEFIPS when FIPS mode is enabled.

Default: Disabled

Dynamically update the runtime when SSL configuration changes occur:

Specifies that all of the SSL-related attributes and LTPA keys that change must be read from the configuration dynamically after they have been saved, then reused for new connections. To avoid customer impact, it is recommended that changes to production servers be made during off-peak periods.

Default: Enabled

When this option is selected, the configuration is updated each time you configure an SSL communication.

SSL configurations for selected scopes

Use this page to display Secure Socket Layer (SSL) configurations for selected scopes, such as a cell, node, server, or cluster. From this page you can navigate to configuration panels for the following: SSL configurations, dynamic inbound and outbound endpoint SSL configurations, key stores, key sets, key set groups, key managers, and trust managers.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***.

Name:

Specifies the SSL configuration scope, which is derived from the selected object in the hierarchy.

Data type: Text

Direction:

Specifies the direction for which the SSLConfig applies. Inbound refers to any listener port. Outbound refers to outbound end point connections.

Data type: Text

SSL configuration:

Specifies the SSL configuration that is used by requests at this scope.

Data type: Text

Update certificate alias list:

Specifies the certificate aliases contained in the key store for this SSL configuration can be selected from the **Certificate alias in key store** list. You must update the certificate list after choosing a different SSL configuration alias. If you do not update the list, you will save a certificate alias that is not contained in the SSL configuration.

Manage certificates:

Specifies to open the keystore panel for the key store in this SSL configuration, which enables you to manage personal certificates, signers, and certificate requests.

Certificate alias in key store:

Specifies the certificate to use in the key store.

If you select **None**, the Java Secure Sockets Extension (JSSE) key manager determines which certificate is used. If multiple certificates exist in the key store, the key manager might not consistently select the same certificate.

Data type: Text

SSL configurations collection

Use this page to define a list of Secure Sockets Layer (SSL) configurations.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **SSL configurations**.

Table 148. SSL configurations buttons. This table lists the SSL configurations buttons.

Button	Resulting action
New	The Java Secure Socket Extension (JSSE) repertoire is for Java-based SSL communications. You can define a new JSSE configuration that can be used to create an SSLContext, URLStreamHandler, SSLSocketFactory, SSLServerSocketFactory, and so on, using the com.ibm.websphere.ssl.JSSEHelper API.
Delete	Deletes an existing JSSE configuration (administrator only). Be careful that any references to the SSL configuration have been removed prior to deleting this SSL configuration.

Name:

Specifies the unique name of the SSL configuration in the management scope.

SSL configuration settings

Use this page to define Secure Sockets Layer (SSL) configuration properties.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *nodes name***. Under Related items, click **SSL configurations > New**.

Name:

Specifies the unique name of the SSL configuration within the management scope in which it resides. For ways to programmatically access the properties that are configured for this SSL configuration, see the com.ibm.websphere.ssl.JSSEHelper application programming interface (API).

Data type: Text

Trust store name:

Specifies a reference to a specific trust store used by Java Secure Sockets Extension (JSSE). The trust store holds signer certificates that validate the trust of certificates sent by remote connections during an SSL handshake.

Data type: Text
Default: *selected trust store*

Key store name:

Specifies a reference to a specific key store. The key store holds personal certificates that represent the identity of one side of a connection. The public key of this personal certificate is sent to the other side of the connection to establish trust during the handshake. The remote side of the connection needs the root certificate authority (CA) certificate or self-signed public key (signer) to be in the trust store to validate this personal certificate.

Data type: Text
Default: *selected key store*

Get certificate aliases:

Queries the keystore for the aliases of all the personal certificates in the keystore from which to choose.

Default server certificate alias:

Specifies the certificate alias used as the identity for this SSL configuration if one has not been specified elsewhere.

If you select **None**, the Java Secure Sockets Extension (JSSE) key manager determines which certificate is used. If multiple certificates exist in the key store, the key manager might not consistently select the same certificate.

Data type: Text

Default client certificate alias:

Specifies the certificate alias to be used if this configuration is to be used as a client.

If you select **None**, the Java Secure Sockets Extension (JSSE) key manager determines which certificate is used. If multiple certificates exist in the key store, the key manager might not consistently select the same certificate.

Data type: Text

Management scope:

Specifies the scope where this SSL configuration is visible. For example, if you choose a specific node, then the configuration is visible only on that node and on any servers that are part of that node.

Data type: Text

Secure Sockets Layer client certificate authentication

Client software that wants to establish a secure connect to a server by using Secure Socket Layer (SSL) protocol initiates by leveraging SSL protocol or the enhanced protocol called Transport Layer Security (TLS) to perform a SSL handshake with SSL certificates. A personal certificate can represent the server or it can represent a particular client, and is signed by a Certificate Authority (CA) to ensure that the personal certificate is correctly identified.

SSL ensures that the administrator has the CA signer certificate available that is used to sign the personal certificate, and that it is stored in both the client and or the server trusted store. SSL client certificate authentication takes place during the connection handshake by using SSL certificates.

The following events must occur during this process:

- The server side must determine if client authentication is going to take place. The client authentication must be enabled in the SSL configuration of the server and the Common Secure Interoperability version 2 (CSIv2) configuration if Inter-ORB Protocol (IIOP) is used.
- The CSIv2 configuration must take place in global security, not in a security domain.
- The signer certificate of the client must be extracted from the key store of the client and added to the trust store of the server.
- The signer certificate of the server must to be extracted from the key store of the server and added to the trust store of the client.

Configuring a WebSphere server for client authentication

Client certificate authentication occurs if the server side requests that the client side send a certificate. A Websphere server can be configured for client certificate authentication on the SSL configuration. However, if client authentication is needed for IIOP then it must be configured on the CSIv2 configuration.

To configure client certificate authentication on the SSL configuration using the administrative console:

1. Click **Security > SSL certificate and key management > SSL configurations**.
2. Select a SSL configuration.
3. Under Additional Properties, select **Quality of protection (QoP) settings**.
4. Under Client authentication, select **Required**.
5. Click **OK** to save the changes.

Note: You can also use the modifySSLConfig command with the -clientAuthentication flag set to `true` to enable client authentication. See SSLConfigCommands command group for the AdminTask object for more information about this command.

.

To configure client certificate authentication on a CSIv2 inbound connection using the administrative console:

1. Click **Security > Global Security**.
2. Under RMI/IIOP security, select **CSIv2 inbound communications**.
3. In the CSIv2 Transport Layer section, and under Client certificate authentication, select **Required**.
4. Click **OK** to save the changes

Note: You can also use the configureCSIInbound command with the -clientCertAuth flag set to `Required` to enable client authentication on CSIv2. Read SecurityConfigurationCommands command group for the AdminTask object for more information about this command.

If the client side is set up for client authentication, the signer certificate of the client must be added to the trust store of the server. When you have a certificate from the client in a certificate file it can be added to the trust store of the server.

To add a signer to the trust store of the server using the administrative console:

1. Click **Security > SSL certificate and key management > Key stores and certificates**.
2. Select the trust store that is configured for client authentication.
3. Under Additional Properties, select **Signer Certificates**.
4. Click **Add**.
5. In the Alias field, type an alias name under which to store the certificate.
6. In the File name box, type the full path to the certificate file.
7. Click **OK** to save the changes

Note: You can also use the addSignerCertificate command to add a signer to the trust store of the server. Read SignerCertificateCommands command group for the AdminTask object for more information about this command.

Note: If you are using client authentication in a cluster environment, client authentication must be configured for each node that the servers in the cluster are located in.

Setting up the client side for client authentication

Clients:

Administrative clients, thin clients or pure clients must have a personal certificate in their key stores. The WebSphere client default key store that is created when WebSphere Application Server is installed already has a personal certificate in it. This key store can be found in the **ssl.client.props** file in the com.ibm.ssl.keyStore property. The client key stores are not managed by WebSphere Application Server, so the Key Management utility (iKeyman) or Java keytool utility can be used to extract the certificate to a certificate file.

To extract a certificate using iKeyman:

1. Start iKeyman.
2. Select **Key Database File > open**.
3. Enter the path to the keystore file. You can obtain this from the **ssl.client.props** file.
4. Click **OK**.
5. Enter the password to the key store and click **OK**
6. Under Personal Certificates, select the client default certificate.
7. Enter a path and file name for the certificate file and click **OK**.

The file that contains the extracted certificate can be used to add the signer to the trust store of the server. Follow the steps in the "Configuring a WebSphere server for client authentication" section to add that signer to the server trust store.

If the communication is over IIOP, the following properties must be set in the **sas.client.props** file.

- Enable SSL:

  ```
  com.ibm.CSI.performTransportAssocSSLTLSSupported=true
  com.ibm.CSI.performTransportAssocSSLTLSRequired=false
  ```

- Disable client authentication at the message layer:

  ```
  com.ibm.CSI.performClientAuthenticationRequired=false
  com.ibm.CSI.performClientAuthenticationSupported=false
  ```

- Enable client authentication at the transport layer (this is supported, but not required):

  ```
  com.ibm.CSI.performTLClientAuthenticationRequired=false
  com.ibm.CSI.performTLClientAuthenticationSupported=true
  ```

Thin clients and pure clients might not use the WebSphere Application Server SSL properties file, **ssl.client.props**. They most likely use the Java system properties to set the client key store and trust store. The signer certificate of the server must be added to the trust store that is specified with the java.net.ssl.trustStore system property. Keytool or iKeyman can be used to add the signer certificate. The signer must be extracted from the personal certificate in the key store specified by the javax.net.ssl.keyStore system property, and added to the trust store of the server.

For example:

```
javax.net.ssl.keyStore
javax.net.ssl.keyStorePassword
javax.net.ssl.keyStoreType
javax.net.ssl.trustStore
javax.net.ssl.trustStorePassword
javax.net.ssl.trustStoreType
```

Server acting as a client:

The client can be a WebSphere server acting as a client. If so, determine which SSL configuration is being used as the client side of the communication, extract it's certificate's signer and add it to the server side trust store. It is recommended that the root certificate signer be used.

To extract the root certificate using the administrative console:

1. Click **Security > SSL certificate and key management > Key stores and certificates**.

2. Under the Keystore usages pull-down, select **Root certificate keystore**.
3. Select either **DmgrDefaultRootStore** (for a network deployment server) or **NodeDefaultRootStore** (for an application server).
4. Under Additional Properties, select **Personal certificates**.
5. Select the default root certificate (usually called *root*), and then click **Extract**.
6. In the Certificate file name box, type a full path to the file in which to hold the certificate.
7. Click **OK** to save.

Note: You can also use the extractCertificate command to extract the root certificate. Read PersonalCertificateCommands command group for the AdminTask object for more information about this command.

The certificate file that is created can be carried to the server side and added to the trust store of the server.

When a server acts as a client, the client side server requires the signer from the destination server. The signer can be retrieved using the signer certificate **Retrieve from port** option.

To retrieve the signer from the port using the administrative console:
1. Click **Security > SSL certificate and key management > Key stores and certificates**.
2. Select the trust store of the server from the collection.
3. Under Additional Properties, select **Signer certificates**.
4. Click **Retrieve from port**.
5. Enter a destination host name and a destination port name.
6. Enter an alias name for the certificate.
7. Click **Retrieve signer information**.
8. Click **OK** to save.

You can also use the retrieveSignerFromPort command to retrieve the signer from the port. Read SignerCertificateCommands command group for the AdminTask object for more information about this command.

Setting up a browser for client authentication:

When WebSphere Application Server is configured for client certificate authentication, and an attempt is made to access the server from a browser, the browser must have a certificate for the client certificate authentication. If the default SSL configuration of the server was modified to enable client certificate authentication you are unable to login to the administrative console.

You can create a certificate for the browser by using the administrative console. You must first create a key store and then create a chained certificate. After the certificate is created, use the instructions for your browser to import a certificate. Browsers require that each part of the chain be added to verify the certificate, so the root certificate must be extracted and added to the browser. Follow the instructions in the "Setting up the client side for client authentication" section for information about extracting the root certificate.

To create a key store using the administrative console:
1. Click **Security > SSL certificate and key management > Key stores and certificates**.
2. Click **New**.
3. Enter a name for the key store.
4. Enter the full path to the key store file.

5. Enter a password for the key store and then confirm.
6. Click **OK** to save.

To create a chained certificate using the administrative console:
1. Click **Security > SSL certificate and key management > Key stores and certificates**.
2. Select the key store you created previously.
3. Under Additional Properties, click **Personal certificates**.
4. In the pull-down list under the Create button, select **Chained Certificate**.
5. Enter an alias name for the certificate.
6. Provide a common name for the certificate. The name is the "CN=" part of the subject DN.
7. You can enter information in any of the remaining fields to build the subject DN of the chained certificate.
8. Click **OK** to save.

You can also use the createKeyStore command to create a key store. Read KeyStoreCommands command group for the AdminTask object for more information about this command.

You can also use the createChainedCertificate command to create a chained certificate. Read PersonalCertificateCommands command group for the AdminTask object for more information about this command.

Note: When client certificate authentication is enabled, web certificate authentication can then be performed as discussed in the next section.

Web certificate authentication

Certificate base authentication can be performed on Java 2 Platform, Enterprise Edition (J2EE) web modules when the module is configured for client certificate authentication. This enables a user to login to a web module using a certificate to authenticate, and to map that certificate to a user from the registry.

Enabling web certificate authentication requires that the SSL configuration of the server be configured for client certificate authentication on the server where the module is installed.

The server side determines that client authentication is to take place. See the "Configuring a WebSphere server for client authentication" section for information about how to configure client authentication. The client side must have the signer from the server to add to the client truststore. See the "Setting up the client side for client authentication" section for more information.

The **web.xml** file of the web module must have the authentication method set to CLIENT-AUTH in the login-config section of the **web.xml** file:

```
<login-config>
<auth-method>CLIENT-CERT</auth-method>
</login-config>
```

The certificate must map to a user in the registry or you are unable to login to that web module.

For localOS user registries, the CN value of the certificate subject DN must map to a user in the local OS user registry. For example, if the certificate subject DN is `CN=tester,o=ibm,c=us`, then `tester` is the user searched for in the local user registry. If that user does not exist in the local registry then the authentication fails.

The Lightweight Directory Access Protocol (LDAP) user registry provides more options for mapping a certificate to a user identity. The default certificate mapping mode in LDAP is used for an exact DN match between the entry in the LDAP registry and the subject DN in the certificate. For example, if the certificate

DN is `CN=user1,o=ibm,c=us`, then there must be an entry in the LDAP registry with that exact value. The LDAP user registry also has a certificate filter option that can provide a match to a particular part of the certificate subject DN against entries in the LDAP repository. For more detail on LDAP certificate mapping, read "Lightweight Directory Access Protocol repository configuration settings".

A federated repository file-based registry does not support certificate mapping, but the federated repository LDAP registry does. It uses the same mapping rules and properties that the LDAP user registry uses.

Custom user registry can map certificates to a user if the custom registry implemented the mapCertificate() method.

Certificate authority (CA) client configuration

Use this page to create, modify, and configure a certificate authority (CA) client.

To view this administrative console page, click **Security > SSL Configurations and key management** . Under Related Items, click **Certificate Authority (CA) client configurations**. Then click either the **New** button or select an existing CA client by clicking on its <client_name>.

Name:

Specifies the unique name of the CA client configuration. This is the name to identify the CA client object. This name needs to be unique to the scope.

Data type: String

Implementation class:

Specifies the name of the module that implements the com.ibm.wsspi.ssl.WSKPIClient interface that is used to act as a client to a CA. This implementation class connects to the CA server and performs a certificate create, revoke, or replace.

Default: String

CA server host name:

Specifies the host name of the CA server, if the implementation requires a host name.

Data type: String

Port:

Specifies the port where the CA server will communicate, if the implementation requires a port.

Data type: String

User name:

Specifies the user Id used to connect to the CA server, if the implementation requires a user to login to the CA.

Data type: String

Password:

Specifies the password for the connection to the CA server.

Data type: String

Confirm password:

Confirms the password that is provided in the password field.

Data type: String

Number of times to poll:

Specifies the number of times to check the CA server to see if the certificate is complete. This poll number applies to the CA that does not return certificates right away.

Default: 5

Polling interval when requesting certificates:

Specifies the amount of time, in minutes, between checks to the CA server to see if the certificate is complete.

Default: 10

Custom properties:

Specifies arbitrary name and value pairs of data. The name is a property key, and the value is a string value that can be used to set internal system configuration properties.

Data type: string

Certificate authority (CA) client configuration collections

Use this page to define and manage certificate authority (CA) clients or view and modify existing CA clients.

This panel allows you to create a certificate authority (CA) client object in the configuration. You can also view and modify existing CA clients. The information in the CA client object can then be used by the runtime to connect to a CA server to request, revoke, or query a certificate.

To view this administrative console page, click **Security > SSL Configurations and key management >** . Under Related Items, click **Certificate Authority (CA) client configurations**.

Table 149. CA client configuration buttons.

This table describes the CA client configuration buttons.

Button	Resulting action
New	Adds a new CA client object that can be referenced by Secure Sockets Layer (SSL) configurations.
Delete	Deletes an existing CA client object.

Name:

Identifies the unique name of the CA client configuration.

Implementation class:

Identifies the name of the module that implements the **com.ibm.wsspi.ssl.WSKPIClient** interface that is used to act as a client to a CA.

Management Scope:

Identifies the scope where this secure sockets layer (SSL) configuration is visible.

Creating a chained personal certificate in SSL

A chained personal certificate is a personal certificate that is created by using another personal certificate to sign it. This chaining allows a certificate to be signed with a certificate (a root certificate) that has a long life span. Root certificates are stored in the **DmgrDefaultRootStore** or **NodeDefaultRootStore**. The server's default personal certificate is a chained certificate created when the profile is created. Chained certificates can also be created after profile creation

Before you begin

You use the administrative console to create a chained personal certificate.

Procedure

1. Click **Security > SSL certificate and key management**.
2. Under Related Items, click **Key stores and certificates**.
3. Click a **<keystore name>** to which you want to add the chained personal certificate.
4. Under Additional Properties, click **Personal certificates** .
5. Click the **Create** button and select **Chained Certificate** The **listCertificates** AdminTask can be used to generate the list of root certificates available to sign the certificate.
6. Fill in the following information to the General Properties section as follows:
   - Supply an alias name.
   - Select Root certificate from the pull down list.
   - Key size
   - Common name
   - Validity period
   - Organization
   - Organization Unit
   - Locality
   - State/Province
   - Zip code
   - Country or region
   -
7. Click **Apply** then **OK**.

Results

The certificate is created, signed by the root certificate specified, and stored in the keystore. Once a chained personal certificate is created, the certificate can be used by the runtime for SSL communication.

Recovering deleted certificates in SSL

The SSL configuration contains a keystore created to hold personal certificates that were deleted from other keystores in the configuration. Perform this task to recover deleted certificates.

Before you begin

The SSL configuration contains a keystore created to hold personal certificates that were deleted from other keystores in the configuration. On a stand alone application server the keystore is called `NodeDefaultDeletedStore` and on a deployment manager the keystore is called `DmgrDefaultDeletedStore`.

When a personal certificate is deleted from a keystore using the administrative console or in a script using the deleteCertificate AdminTask, a copy of the certificate is stored in the `DmgrDeletedKeyStore` or `NodeDeletedKeyStore`. The personal certificate takes the alias of <keystore>_<alias> > in the deleted keystore. If the alias name is already used in that deleted keystore a <unique number> is appended to the alias.

A personal certificate can be recovered from the deleted keystore by importing or exporting the personal certificate to a keystore in the configuration. To recover a personal certificate using the administrative console perform the following steps:

Procedure

1. Click **Security > SSL certificate and key management**.
2. Under Related Items, click **Key stores and certificates**.
3. From the Keystore usages drop-down list, select "Deleted certificates keystore".
4. Click **DmgrDefaultDeletedStore** or **NodeDefaultDeletedStore**.
5. Under Additional Properties, click **Personal certificates**.
6. Select a certificate.
7. Select **Export**
8. Click **OK**.
9. Perform the following:
   - • Enter the keystore password of the deleted keystore.
   - • Enter The alias to be assigned to the certificate (in the key store that will receive the certificate).
   - • Select the ‘Managed key store’ radio button.
   - • Select the key store from the drop down list that will receive the certificate.
   - Click **Apply** then **OK**.

Results

Note: To recover a personal certificate you can also use the exportCertToManagedKS AdminTask command.

Renewing a certificate in SSL

If a personal certificate has been compromised or is about to expire, then it should be renewed. Renewing a certificate recreates the certificate with all the information from the original certificate, but with a new expiration period and public/private key pair. Only self-signed certificates and chained certificates created by WebSphere can be renewed. If the certificate used to sign the chained certificate is not in the root keystore then the default root certificate is used to renew the certificate.

Before you begin

You use the administrative console to renew the certificate.

Procedure

1. Click **Security > SSL certificate and key management**.
2. Under Related Items, click **Key stores and certificates**.
3. Click the appropriate **<keystore name>** to which you want to add the new certificate.

Note: Only self-signed certificates and chained certificates signed with root certificates from the root keystore can be renewed.

4. Under Additional Properties, click **Personal certificates** to list the personal certificates.
5. Select a personal certificate from the list.
6. Click the **Renew** button.
7. Click **Apply** then **OK**.

Results

The certificate is renewed in the key store selected in the path to this panel . If the certificate is not a self-signed certificate or a chained certificate signed with a root certificate from the default root store, an error is returned.

Note: If this command is used with a CA certificate, an error occurs.

Revoking a CA certificate in SSL

If a certificate authority (CA) certificate is compromised and the servers cannot trust it anymore that CA certificate can be revoked. To revoke a CA certificate, you perform the following task.

Before you begin

You use the administrative console to replace or revoke a CA certificate.

Procedure

1. Click **Security > SSL certificate and key management**.
2. Under Related Items, click **Key stores and certificates**.
3. Click a **<keystore name>** to which you want to add the new CA certificate.
4. Under Additional Properties, click **Personal certificates** to list the personal certificates.
5. Select a certificate to revoke (a CA certificate)
6. Click the **Revoke** button.
7. Fill in the following information to the CA certificate section.
   - Revocation password
   - Revocation reason
8. Click **Apply** then **OK**.

Results

The certificate is revoked in the key store selected in the path. If the certificate selected was not a CA certificate, then an error is returned.

What to do next

Using a CA client to create a personal certificate to be used as the default personal certificate

An external certificate authority (CA) certificate can be used as the server default personal certificate. The CA certificate can be created using a CA client.

Before you begin

What you need to have before you perform this task is as follows:

- A certificate authority (CA) to make the certificate request to.

- A module that implements the **com.ibm.wsspi.ssl.WSPKIClient** interface. This module is needed to connect to the CA server and request a certificate.

You use the administrative console to view or modify a CA client.

Procedure

1. Click **Security > SSL certificate and key management**.
2. Under Related Items, click **Certificate Authority (CA) client configurations**. A panel displaying the existing CA clients appears.
3. Click the **New** button.
4. Enter the CA client information as required.
   - Name of the CA client.
   - The management scope (selected from the drop-down list.
   - Implementation class.
   - CA server host name.
   - User name.
   - Password.
   - Confirm of password.
   - Number of times to poll.
   - Polling interval (in minutes) when requesting certificates.
   - Custom properties.
5. Click **Apply** then **Save**.
6. Navigate to the Server default key store personal certificate. **Security > SSL configuration and certificate management > Key stores and certificates > <server_default_keystore>** . Under Additional properties, click **Personal certificates**
7. Click the **Create** button and select **CA-signed certificate**
8. Fill in the following information to the CA certificate section.
   - Revocation password
   - Confirm password.
   - Select the CA client that applies to this CA certificate.

     Note: You can create a new CA client to apply to this CA authority by clicking the **New** button.
   - Fill in the following information to the Request Specification section:
     – Select the radio button for Predefined request alias if you have a predefined alias.
     – If you do not have a predefined alias, fill in the following fields:
       - Type an alias name in the Alias field. The alias identifies the certificate request in the keystore.
       - Type a common name (CN) value. This value is the CN value in the certificate distinguished name (DN).
       - **Optional:** Type an organization value. This value is the O value in the certificate DN.
       - **Optional:** Select a key size value. The valid key size values are 512, 1024, 2048, 4096, and 8192. The default key size value is 2048 bits.
       - Locality
       - **Optional:** Type the State or Province value. This value is the ST value in the certificate DN.
       - **Optional:** Type a zip code value. The zip code value is the POSTALCODE value in the certificate DN.
       - **Optional:** Type a country or region value from the list. This country value is the C= value in the certificate request DN.

- Validity period

9. Click **Apply** then **Save**.
10. Navigate to the Server Default Key store's personal certificates **Security > SSL configuration and certificate management > Key stores and certificates > <server_default_keystore>** . Under Additional properties, click **Personal certificates**
11. Select the server default personal certificate and click the **Replace** button.
12. Select the CA certificate alias from the list of aliases.
13. Click **Apply** then **Save**.

Results

The CA certificate alias replaces the alias of the default certificate in places where it is referenced in the configuration. All signer certificates from the default certificate are replaced with the signer certificate from the CA certificate.

Creating a CA certificate in SSL

Certificates can be created by a certificate authority (CA) when a CAClient object is configured to connect to the CA to create the certificate. Certificates created by a certificate authority (CA) with a CA client are tracked in the security configuration in an object called `CACertificate`. The certificate is stored in a keystore and a `CACertificate` object is added to the configuration to reference the certificate. CA certificates are personal certificates.

Before you begin

Before you begin, a CA client must be created to connect to the CA server. You then use the administrative console to create a CA certificate.

Note: In this release of WebSphere Application Server, the valid key size values are 512, 1024, 2048, 4096, and 8192. The default key size value is 2048 bits.

Procedure

1. Click **Security > SSL certificate and key management**.
2. Under Related Items, click **Key stores and certificates**.
3. Click a **<keystore name>** to which you want to add the new CA certificate.
4. Under Additional Properties, click **Personal certificates** to create a new CA certificate in the configuration.

   Note: You can also create a CA certificate by using the **requestCACertificate** AdminTask .
5. Click the **Create** button and select **CA-signed Certificate**
6. Fill in the following information to the CA certificate section.
   - Revocation password
   - Confirm password.
   - Select the CA client from the pull down list.

     Note: You can create a new CA client to apply to this CA authority by clicking the **New** button.
   - Fill in the following information to the Request Specification section:
     – Select the radio button for a predefined request alias if a certificate request is already created.
     – If you do not have a predefined certificate request alias, fill in the following fields:
       a. Type an alias name in the Alias field. The alias identifies the certificate request in the keystore.

b. Type a common name (CN) value. This value is the CN value in the certificate distinguished name (DN).
c. **Optional:** Type an organization value. This value is the O value in the certificate DN.
d. **Optional:** Select a key size value. The valid key size values are 512, 1024, 2048, 4096, and 8192. The default key size value is 2048 bits.
e. Locality
f. **Optional:** Type the State or Province value. This value is the ST value in the certificate DN.
g. **Optional:** Type a zip code value. The zip code value is the POSTALCODE value in the certificate DN.
h. **Optional:** Type a country or region value from the list. This country value is the C= value in the certificate request DN.

7. Click **Apply** then **OK**.

Results

The certificate is stored in the keystore selected in the path to this panel and a CACertificate configuration object is created. Once a CA certificate is created the certificate can be used by the runtime for SSL communication.

An existing certificate request can be used to create the CA certificate or a new certificate request can be created. This panel uses the requestCAClient AdminTask to create the CA certificate.

Developing the WSPKIClient interface for communicating with a certificate authority

Implementing the WSPKIClient interface enables WebSphere Application Server security to communicate with a remote certificate authority (CA).

Procedure

1. Initialize the WSPKIClient method, with init(java.util.HashMap).

   ```
   public void init(java.util.HashMap initAttrs) throws WSPKIException;
   ```

   This method is called by WebSphere Application Server runtime to set up connection information to a CA.

2. • Request a certificate with requestCertificate(byte[], X500Principal, byte[], java.util.HashMap).

   ```
   public X509Certificate[] requestCertificate(byte[] certReq,
   X500Principal SubjectDN, byte[] revocationPassword,
   java.util.HashMap customAttrs) throws WSPKIException;
   ```

   This method is called by WebSphere Application Server runtime to connect to a CA and requests a certificate signed by the authority. A X509Certificate[] is returned if the requested certificate is created. If a null is returned then queryCertificate() is called to check if the certificate is ready. This method is used when the CA requires manual intervention to process a certificate request.

   You can invoke this operation from the administrative console using the “Creating a CA certificate in SSL” on page 1759 task and from a client using the requestCertificate script.

3. • Revoke a certificate with revokeCertificate(X509Certiifcate[], byte[], String, java.util.HashMap).

   ```
   public void revokeCertificate(X509Certificate[] cert, byte[] revocationPassword,
   String revocationReason, java.util.HashMap customAttrs) throws WSPKIException;
   ```

   This method called by WebSphere Application Server runtime to submit a request to a CA to revoke a certificate.

   You can invoke this operation from the administrative console using the revoke CA certificate task, “Revoking a CA certificate in SSL” on page 1757, or using the revokeCertificate script.

4. • Query a certificate with queryCertificate(X509Certiifcate[], byte[], java.util.HashMap).

```
public X509Certificate[] queryCertificate(byte[] certReq,
java.util.HashMap customAttrs) throws WSPKIException;
```

This method is called by WebSphere Application Server runtime to query if certificate creation is completed on the CA. A X509Certificate[] is returned if certificate request is complete. A null is returned if the certificate request is pending.

You perform this operation from the administrative console using the Query (link to usec_sslperscertreqs.html) option, see “Personal certificate requests collection” on page 1811and from a client using the queryCertificate script.

Results

the WSPKIClient interface for communicating with a certificate authority (CA) is implemented.

Creating a custom trust manager configuration for SSL

You can create a custom trust manager configuration at any management scope and associate the new trust manager with a Secure Sockets Layer (SSL) configuration.

Before you begin

You must develop, package, and locate a Java Archive JAR file for a custom key manager in the `was.install.root/lib/ext` directory on WebSphere Application Server. For more information, see “Example: Developing a custom trust manager for custom SSL trust decisions” on page 1765.

About this task

Complete the following steps in the administrative console:

Procedure

1. Decide whether you want to create the custom trust manager at the cell scope or below the cell scope at the node, server, or cluster, for example.

   Important: When you create a custom trust manager at a level below the cell scope, you can associate it only with a Secure Sockets Layer (SSL) configuration at the same scope or higher. An SSL configuration at a scope lower than the trust manager does not see the trust manager configuration.

   - To create a custom trust manager at the cell scope, click **Security > SSL certificate and key management > Trust managers**. Every SSL configuration in the cell can select the trust manager at the cell scope.
   - To create a custom trust manager at a scope below the cell level, click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Trust managers**.
2. Click **New** to create a new custom trust manager.
3. Type a unique trust manager name.
4. Select the **Custom** implementation setting. The custom setting enables you to define a Java class with an implementation of the javax.net.ssl.X509TrustManager Java interface and, optionally, the com.ibm.wsspi.ssl.TrustManagerExtendedInfo WebSphere Application Server interface.

   Note: The standard implementation setting applies only when the trust manager is already defined in the Java security provider list as a provider and an algorithm, which is not the case for a custom trust manager.
5. Type a class name, for example, com.ibm.test.CustomTrustManager.
6. Select one of the following actions:

- Click **Apply**, then click **Custom properties** under Additional Properties to add custom properties to the new custom trust manager. When you are finished adding custom properties, click **OK** and **Save**, then go to the next step.
- Click **OK** and **Save**, then go to the next step.

7. Click **SSL certificate and key management** in the page navigation at the top of the panel.
8. Select one of the following actions:
   - Click **SSL configurations** under Related Items for a cell-scoped SSL configuration.
   - Click **Manage endpoint security configurations** to select an SSL configuration at a lower scope.
9. Click the link for the existing SSL configuration that you want to associate with the new custom trust manager. You can create a new SSL configuration instead of associating the custom trust manager with an existing configuration. For more information, see "Creating a Secure Sockets Layer configuration" on page 1741.
10. Click **Trust and Key managers** under Additional Properties. If the new custom trust manager is not listed in the **Additional ordered trust managers** list, verify that you selected an SSL configuration scope that is at the same level or below the scope that you selected in Step 8.
11. Click **Add**. This action adds the new trust manager to the list of custom trust managers.
12. Click **OK** and **Save**.

Results

You have created a custom trust manager configuration that references a JAR file in the install directory of WebSphere Application Server and associates it with an SSL configuration during the connection handshake.

What to do next

You can create a custom trust manager for a pure client. For more information, see the TrustManagerCommands command group for the AdminTask object topic.

Trust and key managers settings:

Use this page to specify trust and key managers for the selected SSL configuration.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under **Related items** click **SSL configurations >** *SSL_configuration_name*. Under **Additional Properties** click **Trust and key managers**.

Attention: The application server checks the default trust managers first before checking the additional ordered trust managers in descending order.

Default trust manager:

Specifies the default trust manager. The default trust manager is IbmPKIX, which can be selected when certificate revocation checks must be made using the X509Certificate CRL distribution list. The other default trust manager is IbmX509.

Data type:	Text
Default:	ibmPKIX

Additional ordered trust managers:

Specifies additional trust managers that are used in the order shown for this SSL configuration.

Add:

Specifies to add the selection to the **Additional ordered trust managers** right-hand list.

Remove:

Specifies to remove the selection from the **Additional ordered trust managers** right-hand list.

Key manager:

Specifies the key manager that runs for this SSL configuration.

Data type: Text
Default: IbmX509

Trust managers collection:

Use this page to define the implementation settings for the trust manager. A trust manager is a class that is invoked during a Secure Sockets Layer (SSL) handshake to make trust decisions about the remote end point. A default trust manager is used to validate the signature and expiration of the certificate. Custom trust managers can be plugged in to perform an extended certificate and host name check.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Trust managers**.

Table 150. Trust managers buttons. This table describes the trust manager buttons.

Button	Resulting action
New	Adds a new trust manager that can be selected by an SSL configuration. A trust manager is invoked during an SSL handshake and can decide whether the handshake should be accepted based on the information it knows about the remote certificate and host.
Delete	Deletes an existing trust manager. Make sure the trust manager is not referenced by any SSL configuration before you delete it.

Name:

Specifies the name of the trust manager. This name is used as a selection in the SSL configuration panel.

Class name:

Specifies a class that implements the javax.net.ssl.X509TrustManager interface. Optionally, the class can implement the com.ibm.wsspi.ssl.TrustMangerExtendedInfo interface to get extended information about the connection. The class can use the information to verify the host name and so on.

Algorithm:

Specifies the algorithm name of the trust manager that is implemented by the selected provider.

Trust managers settings:

This page enables you to view and set definitions for trust manager implementation settings. A trust manager is a class that gets invoked during a Secure Sockets Layer (SSL) handshake to make trust decisions about the remote end point. A default trust manager is used to validate the signature and expiration of the certificate. Custom trust managers can be plugged in to perform an extended certificate and hostname check.

To view this administrative console page, click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration*** . Under Related items click **Trust managers > New** .

Name:

Specifies the name of the trust manager.

Data type: Text
Default: ibmX509TrustManager

Management scope:

Specifies the scope where this Secure Sockets Layer (SSL) configuration is visible. For example, if you choose a specific node, then the configuration is only visible on that node and any servers that are part of that node.

This field is not editable and provides information only.

Standard:

Specifies that the trust manager selection is available from a Java provider that is installed in the java.security file. This provider might be shipped by the Java Secure Sockets Extension (JSSE) or might be a custom provider that implements the javax.net.ssl.X509TrustManager interface.

Default: Enabled

Provider:

Specifies the provider name that has an implementation of the javax.net.ssl.X509TrustManager interface. This provider is typically set to IBMJSSE2.

Enabled when **Standard** is selected.

Default IBMJCE

Algorithm:

Specifies the algorithm name of the trust manager implemented by the selected provider.

Enabled when **Standard** is selected.

Default ibmX509 or IbmPKIX
Range ibmX509, IbmPKIX

Custom:

Specifies that the trust manager selection is based on a custom implementation class that implements the javax.net.ssl.X509TrustManager interface and optionally the com.ibm.wsspi.ssl.TrustManagerExctendedInfo interface to obtain additional connection information that is not otherwise available.

Default: Disabled

Class name:

Specifies a class that implements the javax.net.ssl.X509TrustManager interface. Optionally, the class can implement the com.ibm.wsspi.ssl.TrustMangerExtendedInfo interface to get extended information about the connection. The class can use the information to verify the host name and so on.

Enabled when **Custom** is selected.

Data type: Text

Example: Developing a custom trust manager for custom SSL trust decisions:

The following example is of a sample custom trust manager. The custom trust manager makes no trust decisions but instead uses the information in the X.509 certificate that it references to make decisions.

After you build and package the custom trust manager, configure it either from the `ssl.client.props` file for a pure client or the SSLConfiguration TrustManager link in the administrative console. See "Trust manager control of X.509 certificate trust decisions" on page 1709 for more information about trust managers.

Note: This example should only be used as a sample, and is not supported.

```
import java.security.cert.X509Certificate;
import javax.net.ssl.*;
import com.ibm.wsspi.ssl.TrustManagerExtendedInfo;

public final class CustomTrustManager implements X509TrustManager,
TrustManagerExtendedInfo
{
    private static ThreadLocal threadLocStorage = new ThreadLocal();
    private java.util.Properties sslConfig = null;
    private java.util.Properties props = null;

    public CustomTrustManager()
    {
    }

    /**
     * Method called by WebSphere Application Server run time to set the target
     * host information and potentially other connection info in the future.
     * This needs to be set on ThreadLocal since the same trust manager can be
     * used by multiple connections.
     *
     * @param java.util.Map - Contains information about the connection.
     */
    public void setExtendedInfo(java.util.Map info)
    {
        threadLocStorage.set(info);
    }

    /**
     * Method called internally to retrieve information about the connection.
     *
     * @return java.util.Map - Contains information about the connection.
     */
    private java.util.Map getExtendedInfo()
    {
        return (java.util.Map) threadLocStorage.get();
    }

    /**
     * Method called by WebSphere Application Server run time to set the custom
     * properties.
     *
     * @param java.util.Properties - custom props
     */
    public void setCustomProperties(java.util.Properties customProps)
    {
        props = customProps;
    }

    /**
     * Method called internally to the custom properties set in the Trust Manager
     * configuration.
     *
     * @return java.util.Properties - information set in the configuration.
     */
    private java.util.Properties getCustomProperties()
    {
        return props;
```

```
    }

    /**
     * Method called by WebSphere Application Server runtime to set the SSL
     * configuration properties being used for this connection.
     *
     * @param java.util.Properties - contains a property for the SSL configuration.
     */
    public void setSSLConfig(java.util.Properties config)
    {
        sslConfig = config;
    }

    /**
     * Method called by TrustManager to get access to the SSL configuration for
     * this connection.
     *
     * @return java.util.Properties
     */
    public java.util.Properties getSSLConfig ()
    {
        return sslConfig;
    }

    /**
     * Method called on the server-side for establishing trust with a client.
     * See API documentation for javax.net.ssl.X509TrustManager.
     */
    public void checkClientTrusted(X509Certificate[] chain, String authType)
        throws java.security.cert.CertificateException
    {
        for (int j=0; j<chain.length; j++)
        {
            System.out.println("Client certificate information:");
            System.out.println("  Subject DN: " + chain[j].getSubjectDN());
            System.out.println("  Issuer DN: " + chain[j].getIssuerDN());
            System.out.println("  Serial number: " + chain[j].getSerialNumber());
            System.out.println("");
        }
    }


    /**
     * Method called on the client-side for establishing trust with a server.
     * See API documentation for javax.net.ssl.X509TrustManager.
     */
    public void checkServerTrusted(X509Certificate[] chain, String authType)
        throws java.security.cert.CertificateException
    {
        for (int j=0; j<chain.length; j++)
        {
            System.out.println("Server certificate information:");
            System.out.println("  Subject DN: " + chain[j].getSubjectDN());
            System.out.println("  Issuer DN: " + chain[j].getIssuerDN());
            System.out.println("  Serial number: " + chain[j].getSerialNumber());
            System.out.println("");
        }
    }

    /**
     * Return an array of certificate authority certificates which are trusted
     * for authenticating peers. You can return null here since the IbmX509
     * or IbmPKIX will provide a default set of issuers.
     *
     * See API documentation for javax.net.ssl.X509TrustManager.
     */
    public X509Certificate[] getAcceptedIssuers()
    {
        return null;
    }
}
```

Creating a custom key manager for SSL

You can create a custom key manager configuration at any management scope and associate the new key manager with a Secure Sockets Layer (SSL) configuration.

Before you begin

You must develop, package, and locate a Java Archive (.JAR) file for a custom key manager in the `was.install.root/lib/ext` directory on WebSphere Application Server.

About this task

Complete the following steps in the administrative console:

Procedure

1. Decide whether you want to create the custom key manager at the cell scope or below the cell scope at the node, server, or cluster, for example.

   Important: When you create a custom key manager at a level below the cell scope, you can associate it only with a Secure Sockets Layer (SSL) configuration at the same scope or higher. An SSL configuration at a scope lower than the key manager does not see the key manager configuration.

   - To create a custom key manager at the cell scope, click **Security > SSL certificate and key management > Key managers**. Every SSL configuration in the cell can select the key manager at the cell scope.
   - To create a custom key manager at a scope below the cell level, click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *SSL_configuration* > Key managers**.
2. Click **New** to create a new key manager.
3. Type a unique key manager name.
4. Select the **Custom** implementation setting. With the custom setting, you can define a Java class that has an implementation on the Java interface javax.net.ssl.X509KeyManager and, optionally, the com.ibm.wsspi.ssl.KeyManagerExtendedInfo WebSphere Application Server interface. The standard implementation setting applies only when the key manager is already defined in the Java security provider list as a provider and an algorithm, which is not the case for a custom key manager. The typical standard key manager is `algorithm = IbmX509, provider = IBMJSSE2`.
5. Type a class name. For example, com.ibm.test.CustomKeyManager.
6. Select one of the following actions:
   - Click **Apply**, then click **Custom properties** under Additional Properties to add custom properties to the new custom key manager. When you are finished adding custom properties, click **OK** and **Save**, then go to the next step.
   - Click **OK** and **Save**, then go to the next step.
7. Click **SSL certificate and key management** in the page navigation at the top of the panel.
8. Select one of the following actions:
   - Click **SSL configurations** under Related Items for a cell-scoped SSL configuration.
   - Click **Manage endpoint security configurations** to select an SSL configuration at a lower scope.
9. Click the link for the existing SSL configuration that you want to associate with the new custom key manager. You can create a new SSL configuration instead of associating the custom key manager with an existing configuration. For more information, see example below.
10. Click **Trust and Key managers** under Additional Properties.
11. Select the new custom key manager in the **Key manager** drop-down list. If the new custom key manager is not listed, verify that you selected an SSL configuration scope that is at the same level or below the scope that you selected in Step 8.
12. Click **OK** and **Save**.

Results

You have created a custom key manager configuration that references a JAR file in the installation directory of WebSphere Application Server and associates the custom configuration with an SSL configuration during the connection handshake.

Example

Developing a custom key manager for custom Secure Sockets Layer key selection. The following example is of a sample custom key manager. This simple key manager returns the configured alias if it is set using the alias properties com.ibm.ssl.keyStoreClientAlias or com.ibm.ssl.keyStoreServerAlias, depending on which side of the connection the key manager is used. The key manager defers to the JSSE default IbmX509 key manager to select an alias if these properties are not set.

After you build and package a custom key manager, you can configure it from either the `ssl.client.props` file for a pure client or by using the SSLConfiguration KeyManager link in the administrative console. See “Key manager control of X.509 certificate identities” on page 1711 for more information about key managers.

Because only one key manager can be configured at a time for any given Secure Sockets Layer (SSL) configuration, the certificate selections on the server side might not work as they would when the default IbmX509 key manager is specified. When a custom key manager is configured, it is up to the owner of that key manager to ensure that the selection of the alias from the SSL configuration supplied is set properly when `chooseClientAlias` or `chooseServerAlias` are called. Look for the `com.ibm.ssl.keyStoreClientAlias` and `com.ibm.ssl.keyStoreServerAlias` SSL properties.

Note: This example should only be used as a sample, and is not supported.

```
package com.ibm.test;

import java.security.cert.X509Certificate;
import com.ibm.wsspi.ssl.KeyManagerExtendedInfo;

public final class CustomKeyManager
 implements javax.net.ssl.X509KeyManager, com.ibm.wsspi.ssl.KeyManagerExtendedInfo
{
    private java.util.Properties props = null;
    private java.security.KeyStore ks = null;
    private javax.net.ssl.X509KeyManager km = null;
    private java.util.Properties sslConfig = null;
    private String clientAlias = null;
    private String serverAlias = null;
    private int clientslotnum = 0;
    private int serverslotnum = 0;

    public CustomKeyManager()
    {
    }

    /**
     * Method called by WebSphere Application Server runtime to set the custom
     * properties.
     *
     * @param java.util.Properties - custom props
     */
    public void setCustomProperties(java.util.Properties customProps)
    {
        props = customProps;
    }

    private java.util.Properties getCustomProperties()
    {
        return props;
    }

    /**
     * Method called by WebSphere Application Server runtime to set the SSL
     * configuration properties being used for this connection.
     *
     * @param java.util.Properties - contains a property for the SSL configuration.
     */
    public void setSSLConfig(java.util.Properties config)
    {
        sslConfig = config;
    }

    private java.util.Properties getSSLConfig()
    {
        return sslConfig;
    }

    /**
     * Method called by WebSphere Application Server runtime to set the default
     * X509KeyManager created by the IbmX509 KeyManagerFactory using the KeyStore
```

```
     * information present in this SSL configuration.  This allows some delegation
     * to the default IbmX509 KeyManager to occur.
     *
     * @param javax.net.ssl.KeyManager defaultX509KeyManager - default key manager for IbmX509
     */
    public void setDefaultX509KeyManager(javax.net.ssl.X509KeyManager defaultX509KeyManager)
    {
        km = defaultX509KeyManager;
    }

    public javax.net.ssl.X509KeyManager getDefaultX509KeyManager()
    {
        return km;
    }

    /**
     * Method called by WebSphere Application Server runtime to set the SSL
     * KeyStore used for this connection.
     *
     * @param java.security.KeyStore - the KeyStore currently configured
     */
    public void setKeyStore(java.security.KeyStore keyStore)
    {
        ks = keyStore;
    }

    public java.security.KeyStore getKeyStore()
    {
        return ks;
    }

/**
     * Method called by custom code to set the server alias.
     *
     * @param String - the server alias to use
     */
    public void setKeyStoreServerAlias(String alias)
    {
        serverAlias = alias;
    }

    private String getKeyStoreServerAlias()
    {
        return serverAlias;
    }

    /**
     * Method called by custom code to set the client alias.
     *
     * @param String - the client alias to use
     */
    public void setKeyStoreClientAlias(String alias)
    {
        clientAlias = alias;
    }

    private String getKeyStoreClientAlias()
    {
        return clientAlias;
    }

    /**
     * Method called by custom code to set the client alias and slot (if necessary).
     *
     * @param String - the client alias to use
     * @param int - the slot to use (for hardware)
     */
    public void setClientAlias(String alias, int slotnum) throws Exception
    {
        if ( !ks.containsAlias(alias))
        {
            throw new IllegalArgumentException ( "Client alias " + alias + "
            not found in keystore." );
        }
        this.clientAlias = alias;
        this.clientslotnum = slotnum;
    }

    /**
     * Method called by custom code to set the server alias and slot (if necessary).
     *
     * @param String - the server alias to use
     * @param int - the slot to use (for hardware)
     */
    public void setServerAlias(String alias, int slotnum) throws Exception
    {
        if ( ! ks.containsAlias(alias))
        {
            throw new IllegalArgumentException ( "Server alias " + alias + "
            not found in keystore." );
```

```
    }
    this.serverAlias = alias;
    this.serverslotnum = slotnum;
}


/**
 * Method called by JSSE runtime to when an alias is needed for a client
 * connection where a client certificate is required.
 *
 * @param String keyType
 * @param Principal[] issuers
 * @param java.net.Socket socket (not always present)
 */
public String chooseClientAlias(String[] keyType, java.security.Principal[]
issuers, java.net.Socket socket)
{
    if (clientAlias != null && !clientAlias.equals(""))
    {
        String[] list = km.getClientAliases(keyType[0], issuers);
        String aliases = "";

        if (list != null)
        {
            boolean found=false;
            for (int i=0; i<list.length; i++)
            {
                aliases += list[i] + " ";
                if (clientAlias.equalsIgnoreCase(list[i]))
                    found=true;
            }

            if (found)
            {
                return clientAlias;
            }

        }
    }

    // client alias not found, let the default key manager choose.
    String[] keyArray = new String [] {keyType[0]};
    String alias = km.chooseClientAlias(keyArray, issuers, null);
    return alias.toLowerCase();
}

/**
 * Method called by JSSE runtime to when an alias is needed for a server
 * connection to provide the server identity.
 *
 * @param String[] keyType
 * @param Principal[] issuers
 * @param java.net.Socket socket (not always present)
 */
public String chooseServerAlias(String keyType, java.security.Principal[]
issuers, java.net.Socket socket)
{
    if (serverAlias != null && !serverAlias.equals(""))
    {
        // get the list of aliases in the keystore from the default key manager
        String[] list = km.getServerAliases(keyType, issuers);
        String aliases = "";

        if (list != null)
        {
            boolean found=false;
            for (int i=0; i<list.length; i++)
            {
                aliases += list[i] + " ";
                if (serverAlias.equalsIgnoreCase(list[i]))
                    found = true;
            }

            if (found)
            {
                return serverAlias;
            }
        }
    }

    // specified alias not found, let the default key manager choose.
    String alias = km.chooseServerAlias(keyType, issuers, null);
    return alias.toLowerCase();
}

public String[] getClientAliases(String keyType, java.security.Principal[] issuers)
{
    return km.getClientAliases(keyType, issuers);
}
```

```
    public String[] getServerAliases(String keyType, java.security.Principal[] issuers)
    {
        return km.getServerAliases(keyType, issuers);
    }

    public java.security.PrivateKey getPrivateKey(String s)
    {
        return km.getPrivateKey(s);
    }

    public java.security.cert.X509Certificate[] getCertificateChain(String s)
    {
        return km.getCertificateChain(s);
    }

    public javax.net.ssl.X509KeyManager getX509KeyManager()
    {
        return km;
    }

}
```

What to do next

You can create a custom key manager for a pure client. For more information, see the keyManagerCommands command group for the AdminTask object.

Associating a Secure Sockets Layer configuration dynamically with an outbound protocol and remote secure endpoint

After you create a Secure Sockets Layer (SSL) configuration, you must associate a secure outbound management scope with the new configuration. In this release, you can associate one SSL configuration with one remote secure endpoint and a different SSL configuration to another remote secure endpoint. Both endpoints can use the same outbound protocol, if appropriate. This task describes how to create the association dynamically.

Before you begin

Dynamic outbound selection requires that you provide only the outbound protocol name, the target host, and the target port so that WebSphere Application Server can make a connection between the SSL configuration and the outbound protocol or remote secure endpoint. The dynamic outbound selection method takes precedence over other selection methods, such as central management and direct selection, but is second to the programmatic method, that is, setting an SSL configuration on the running thread. For more information about the selection types and precedence rules, see “Secure communications using Secure Sockets Layer (SSL)” on page 1700.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > Outbound**.
2. Select the management scope that you want to associate with an SSL configuration on the topology tree.
3. Under Related Items, click **Dynamic outbound endpoint SSL configurations**. The default dynamic outbound configuration name, the target protocol, host, and port connection information, and the SSL configuration name display.
4. Click **New** to create a new dynamic outbound configuration.
5. Type a dynamic outbound configuration name. Use a name that is descriptive of the purpose of the dynamic selection configuration.
6. Optionally, type a dynamic selection configuration description.

7. Type the connection information that you want to associate with the configuration that is displayed in the SSL configuration drop-down list. The connection information must be in the format *protocol name*, *target host*, *target port*. You can substitute an asterisk (*) for any value, as in the following examples, where 443 is a port, www.mycompany.com is a host, HTTP is a protocol, and .hometown.mycompany.com is a target host. You can add multiple connections, but each additional connection can affect outbound performance.
   - ***,*,443**
   - ***,www.mycompany.com,443**
   - **HTTP,.hometown.mycompany.com,***
   - ***,*,***

     gotcha: Do not use this configuration because it matches all outbound specifications. Therefore, no other SSL configuration is used for outbound connections.

   gotcha:
   - Unless the intention is to set the protocol property through the JSSEHelper API, the protocol filter should be set to * (as in the first two examples). See "Dynamic Selection" in “Secure communications using Secure Sockets Layer (SSL)” on page 1700 for more information.
   - The connection protocols that are used for dynamic outboud SSL configuration selection, that are illustrated in the preceding examples, which are not corresponding the protocol name of the URL. To use one of these protocols from a user-written application, programmatic SSL configuration selection must be implemented.
8. Click **Add** to add the new connection to the set of SSL configuration connections. To remove a connection, select it and click **Remove**.
9. Select an SSL configuration from the list.
10. Click **Get certificate aliases** to refresh the certificate aliases that are contained in the associated key store.
11. Choose a certificate alias from the list.
12. Click **OK** and **Save**.

Results

WebSphere Application Server is ready to connect one or more SSL configurations to one or more remote secure endpoints.

What to do next

You can return to the outbound tree and select another management scope to associate with the same or a new outbound configuration.

Programmatically specifying an outbound SSL configuration using JSSEHelper API:

WebSphere Application Server provides a way to specify programmatically which Secure Sockets Layer (SSL) configurations to use prior to making an outbound connection. The com.ibm.websphere.ssl.JSSEHelper interface provides a complete set of application programming interfaces (APIs) for handling SSL configurations.

About this task

Perform the following steps for your application when using the JSSEHelper API to establish an SSL properties object on the thread for use by the runtime. Some of these APIs have Java 2 Security permission requirements. See the JSSEHelper API documentation for more information about the permissions required by your application.

Select the approach that best fits your connection situation when you specify programmatically which Secure Sockets Layer (SSL) configurations to use prior to making an outbound connection.

Procedure

1. Obtain an instance of the JSSEHelper API.

```
com.ibm.websphere.ssl.JSSEHelper jsseHelper  = com.ibm.websphere.ssl.JSSEHelper.getInstance();
```

2. Obtain SSL properties from the WebSphere Application Server configuration or use those provided by your application. Use one of the following options.
   - By direction selection of an alias name, within the same management scope or higher as in the following example:

```
try
{  String alias = "NodeAServer1SSLSettings";
  // As specified in the WebSphere SSL configuration  Properties
 sslProps = jsseHelper.getProperties(alias); }
 catch (com.ibm.websphere.ssl.SSLException e)
 {  e.printStackTrace();   // handle exception }
```

   - By using the getProperties API for programmatic, direction, dynamic outbound, or management scope selection (based on precedence rules and inheritance). The SSL runtime uses the getProperties API to determine which SSL configuration to use for a particular protocol. This decision is based on both the input (sslAlias and connectionInfo) and the management scope from which the property is called. The getProperties API makes decisions in the following order:
     a. The API checks the thread to see if properties already exist.
     b. The API checks for a dynamic outbound configuration that matches the ENDPOINT_NAME, REMOTE_HOST, and or REMOTE_PORT.
     c. The API checks to see if the optional sslAlias property is specified. You can configure any protocol as direct or centrally managed. When a protocol is configured as direct, the sslAlias parameter is `null`. When a protocol is configured as centrally managed, the sslAlias parameter is also `null`.
     d. If no selection has been made, the API chooses the dynamic outbound configuration based on the management scope it was called from. If the dynamic outbound configuration is not defined in the same scope, it then searches the hierarchy to locate one.

     The last choice is the cell-scoped SSL configuration (in WebSphere Application Server, Network Deployment) or the node-scoped SSL configuration (in Base Application Server). The com.ibm.websphere.ssl.SSLConfigChangeListener parameter is notified when the SSL configuration that is chosen by a call to the getProperties API changes. The protocol can then call the API again to obtain the new properties as in the following example:

```
try {  String sslAlias = null;
 // The sslAlias is not specified directly at this time.  String host = "myhost.austin.ibm.com";
 // the target host  String port = "443";
 // the target port   HashMap connectionInfo = new HashMap();
 connectionInfo.put(JSSEHelper.CONNECTION_INFO_DIRECTION,    JSSEHelper.DIRECTION_OUTBOUND);
 connectionInfo.put(JSSEHelper.CONNECTION_INFO_REMOTE_HOST, host);
 connectionInfo.put(JSSEHelper.CONNECTION_INFO_REMOTE_PORT,    Integer.toString(port));
 connectionInfo.put(JSSEHelper.CONNECTION_INFO_ENDPOINT_NAME,    JSSEHelper.ENDPOINT_IIOP);
             java.util.Properties props = jsseHelper.getProperties(sslAlias,    connectionInfo, null); }
 catch (com.ibm.websphere.ssl.SSLException e)
 {  e.printStackTrace();   // handle exception }
```

   - By creating your own SSL properties and then passing them to the runtime, as in the following example:

```
try {
 // This is the recommended "minimum" set of SSL properties. The trustStore can
  // be the same as the keyStore.  Properties sslProps = new Properties();
 sslProps.setProperty("com.ibm.ssl.trustStore", "some value");
 sslProps.setProperty("com.ibm.ssl.trustStorePassword", "some value");
```

```
 sslProps.setProperty("com.ibm.ssl.trustStoreType", "some value");
 sslProps.setProperty("com.ibm.ssl.keyStore", "some value");
 sslProps.setProperty("com.ibm.ssl.keyStorePassword", "some value");
 sslProps.setProperty("com.ibm.ssl.keyStoreType", "some value");
  jsseHelper.setSSLPropertiesOnThread(sslProps); }
 catch (com.ibm.websphere.ssl.SSLException e)
{  e.printStackTrace();   // handle exception }
```

3. Use the JSSEHelper.setSSLPropertiesOnThread(props) API to set the Properties object on the thread so that the runtime picks it up and uses the same JSSEHelper.getProperties API. You can also obtain properties from the thread after they are set with the jsseHelper.getSSLPropertiesOnThread() API, as in the following example:

```
try
{  Properties sslProps = jsseHelper.getProperties(null,    connectionInfo, null);
jsseHelper.setSSLPropertiesOnThread(sslProps); }
 catch (com.ibm.websphere.ssl.SSLException e)
{  e.printStackTrace();   // handle exception }
```

4. When the connection is completed, you must clear the SSL properties from the thread by passing the `null` value to the setPropertiesOnThread API.

```
try
 {  jsseHelper.setSSLPropertiesOnThread(null); }
 catch (com.ibm.websphere.ssl.SSLException e)
 {  e.printStackTrace();   // handle exception }
```

Associating Secure Sockets Layer configurations centrally with inbound and outbound scopes:

After you create a Secure Sockets Layer (SSL) configuration, you must associate a secure inbound or outbound management scope with the new configuration. You can manage the association centrally so that you can easily make changes that affect all the scopes that are lower on the topology and that are associated with the configuration. Beginning with WebSphere Application Server version 6.1, the recommended and the default configuration method is centrally managed SSL configurations.

Before you begin

You can simplify the number of associations that you need to make for an SSL configuration by associating the configuration with the highest level management scope requiring a unique configuration. SSL configuration associations manifest inheritance behaviors. Because of the inheritance behaviors, all of the scopes that are lower on the topology inherit this SSL configuration. For example, an association you make at the cell level affects nodes, servers, clusters, and endpoints. For more information, see “Central management of SSL configurations” on page 1718.

A precedence rule determines which SSL configuration association is used at a particular scope. The highest precedence is given to endpoints on the topology. If you establish an association at the endpoint, this association overrides any prior association that you made higher up on the management scope topology.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management**.
2. Select the **Dynamically update the runtime when SSL configuration changes** check box if you want changes that you make to an existing SSL configuration to occur dynamically. All outbound SSL communications honor the dynamic SSL changes. Protocols that do not use the channel frameworks SSL channel for inbound communications, including Object Request Broker (ORB) and administrative SOAP protocols, do not honor dynamic updates. For more information, see “Dynamic configuration updates in SSL” on page 1735.
3. Click **Manage endpoint security configurations**.
4. Select either the inbound or the outbound tree. After finishing the selected tree, you can return to this step to repeat the following steps for the other tree.

5. Click the link for the selected cell, node, node group, server, cluster, or endpoint on the topology tree. If the scope already has an associated SSL configuration and alias, these objects display in parentheses immediately following the scope name, for example: Node01(NodeDefaultSSLSettings,default). If the deployment manager has federated a node, the node scope SSL configuration overrides the cell scope configuration above it in the topology.
6. Decide whether to override the inherited values that display in the read-only fields. Read-only fields include the management scope name, the direction, and the inherited SSL configuration name and certificate alias.
   - If you are satisfied with these values, do not override them.
   - If you want to override the inherited values, select the **Override inherited values** check box.
7. Select an SSL configuration from the list.
8. Click **Update certificate alias list**. The certificate alias list comes from the key store that is referenced by the new SSL configuration.
9. Click **Manage certificates** if you want to manage the personal certificates that are contained in the key store that is referenced in the SSL configuration.
10. Click **Update certificate alias list** to refresh the list of aliases.
11. Select a certificate alias in the key store to represent the identity of the endpoint.
12. Click **OK** to save your changes.
13. Click **Manage endpoint security configurations and trust zones** to return to the topology tree.
14. Configure the opposite direction on the topology tree using the steps in this task. You can also select additional scopes to associate with the SSL configuration, as needed.

Results

Each SSL configuration at the selected scope and at scopes beneath it on the topology tree have the same SSL configuration properties. The following SSL configuration methods override the centrally managed configurations that you associate in the tree view:

- Direct selection at the endpoint
- Dynamic outbound SSL configuration associations
- Programmatic specifications

What to do next

At any management scope, you can configure the following objects: dynamic outbound endpoint SSL configurations, key stores, key sets, key set groups, key managers, and trust managers. Like SSL configurations, these objects are scoped automatically so that they are not visible higher up in the tree nor are they loaded during runtime by processes that are higher up in the tree.

Selecting an SSL configuration alias directly from an endpoint configuration:

You can associate a secure outbound endpoint with a new Secure Sockets Layer (SSL) configuration directly. If you are migrating from a release prior to version 6.1, WebSphere Application Server still supports configurations that were selected directly at an endpoint. Direct selection always overrides centrally managed configurations and preserves migrated configurations.

About this task

Select an SSL configuration alias directly at the following endpoints:

- **Security > Global security > RMI/IIOP security > CSIv2 outbound transport**
- **Security > Global security > RMI/IIOP security > CSIv2 inbound transport**
- **System administration > Deployment manager > Transport Chain > WCInboundAdminSecure > SSL inbound channel (SSL_1)**

- **System administration > Deployment manager > Administration Services > JMX connectors > SOAPConnector > Custom Properties > sslConfig**
- **System administration > Node agents > nodeagent > Administration Services > JMX connectors > SOAPConnector > Custom Properties > sslConfig**
- **Servers > Application servers > server1 > Messaging engine inbound transports > InboundSecureMessaging > SSL inbound channel (SIB_SSL_JFAP)**
- **Servers > Application servers > server1 > WebSphere MQ link inbound transports > InboundSecureMQLink > SSL inbound channel (SIB_SSL_MQFAP)**
- **Servers > Application servers > server1 > SIP Container Settings > SIP container transport chains > SIPCInboundDefaultSecure > SSL inbound channel (SSL_5)**
- **Servers > Application servers > server1 > Web Container Settings > Web container transport chains > WCInboundAdminSecure > SSL inbound channel (SSL_1)**
- **Servers > Application servers > server1 > Web Container Settings > Web container transport chains > WCInboundDefaultSecure > SSL inbound channel (SSL_2)**

Attention: The central management of SSL configurations can be a more efficient strategy because multiple configurations can be contained within a single SSLConfigGroup. If you need to convert configuration references that are already directly managed to centrally managed configurations, modify each endpoint individually. Use the **AdminConfig.modify** command to set the `sslConfigAlias` value to an empty string (""). Below is an example of doing this:

- Using Jacl:

```
set s1 [$AdminConfig getid /Cell:mycell/Node:mynode/Server:server1/]
set sslChannel [lindex [$AdminConfig list SSLInboundChannel $s1] 0]
$AdminConfig modify $sslChannel [list[list sslConfigAlias ""]]
```

For more information on using this command, see the information about configuring processes using scripting.

For more information on specific wsadmin commands that affect a repertoire as opposed to individual endpoints, see the SSLConfigGroupCommands group for the AdminTask topic.

Complete the following steps in the administrative console:

Note: These steps provide an example to follow when you directly select any of the endpoints listed above.

Procedure

1. Click **Security > Global security > RMI/IIOP security > CSIv2 outbound transport**.
2. Click **Use specific SSL alias**. When you identify a specific SSL alias, you override the centrally managed scope associations.
3. Select an SSL configuration alias from the drop-down list.
4. Click **OK**.
5. Repeat these steps for additional protocols or endpoints, if desired.

Results

By associating the endpoint directly, you have overridden a centrally managed SSL configuration.

What to do next

If you decide to use management scopes instead of endpoints to associate an SSL configuration, follow the steps above, but click **Centrally managed** instead of **Use specific SSL alias**, then click **Manage endpoint security configurations**. The console is redirected to **Security > SSL certificate and key management > Manage endpoint security configurations**.

Enabling Secure Sockets Layer client authentication for a specific inbound endpoint:

When you establish a Secure Sockets Layer (SSL) configuration, you can enable client authentication for a specific inbound endpoint.

Before you begin

The endpoint configuration must already exist in the SSL topology.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > Inbound >** *SSL_configuration*. If you want to enable SSL client authentication for all processes, define an SSL configuration for the new endpoint at the node or cell level so that it is visible to all processes on the same node or on the entire cell. For more information, see "Creating a Secure Sockets Layer configuration" on page 1741.
2. Select **Override inherited values**. The SSL configuration is used for the current scope and any lower scopes that have not already designated an SSL configuration. This field displays for server and node groups within the object hierarchy and does not display for the top-level node or cell.
3. Select an SSL configuration from the drop-down list.
4. Click **Update certificate alias list**.
5. Select a **Certificate alias** from the drop-down list.
6. Click **OK** to save the configuration.

Results

You can repeat the previous steps for each endpoint that uses the same SSL configuration to enable client authentication for the inbound endpoints.

What to do next

CSIv2 Protocol Exception:

The Common Secure Interoperability Version 2 (CSIv2) secure endpoints, used for Remote Method Invocation over the Internet Inter-ORB Protocol (RMI/IIOP) security, cannot override inherited values. While the rest of the SSL properties are effective for CSIv2 when they are selected at the centrally-managed Secure Communications panel, the client authentication selection is controlled by the CSIv2 protocol configuration.

To enable SSL client certificate authentication for the CSIv2 protocol, you must use the CSIv2 inbound and outbound authentication panels. For SSL client authentication to occur between two servers, you must enable (support or require) SSL client certificate authentication for both the inbound and the outbound policies.

WebSphere Application Server can either request (support) clients to provide signer certificates for the SSL handshake, or the server can require clients to provide a valid signer certificate for the SSL handshake, which is a more secure method. However, when the server requires certificates, the server must obtain a signer for each client that connects to the server, which involves more server-side management.

The client certificate should not be used for the identity when it is used from server-to-server. However, when a pure client sends the client certificate it is used for the identity unless a message level identity is specified, such as a user ID or a password.

Do the following to enable client certificate authentication for the CSIv2 protocol for server-to-server:

1. **Click Security > Global security**.
2. Expand the **RMI/IIOP** security section.
3. Click **CSIv2 inbound authentication**.
4. Under Client authentication, select either **supported** or **required**. When you select required, only one SSL port is opened (CSV2_SSL_MUTUALAUTH_LISTENER_ADDRESS). When you select supported, two SSL ports are opened (both CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS and CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS).

   If there are two ports, the client can select either based on the security configuration policy of the port.
5. Click **OK** to save.
6. If you want server-to-server SSL client authentication, then complete the remaining steps. If you don't complete the remaining steps, only pure clients are enabled to send client certificates.
7. Expand the **RMI/IIOP** security section.
8. Click **CSIv2 outbound authentication**.
9. Under Client authentication, select either **supported** or **required**.

The SSL configuration for the inbound secure endpoints for which you enable SSL client certificate authentication must have the signer certificate from any client that attempts to open a connection to that inbound secure endpoint. You must collect those signers and then add them to the trust store associated with the inbound secure endpoints SSL configuration.

Manage endpoint security configurations:

Use this page to select a Secure Socket Layer (SSL) configuration from the Local Topology hierarchy, which includes cells, nodes, node groups, servers, and clusters.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations**.

Local topology:

The Local topology represents the hierarchy of nodes, node groups, clusters, servers, and end points within the cell that comprise a centralized SSL configuration.

The topology acts as a hierarchical tree in terms of inheritance. For example, if an SSL configuration has been associated with a specific node, then all servers within that node will inherit that SSL configuration selection, provided the servers are not associated with an SSL configuration at the server scope. Centralized management of SSL is the default configuration; however, it can be overridden at various locations to directly select a specific SSL alias as in previous releases for backwards compatibility.

Scope	Description
Inbound/Outbound	Specifies the topology tree in terms of connection direction. For example, the inbound tree represents all server endpoints that receive connections at the various servers within the cell. The outbound tree represents the client side of connections from the various servers within the cell.

Scope	Description
Nodes	Specifies the nodes that are part of the cell. The list of nodes is updated anytime a node gets federated into the cell.
Servers	Specifies the servers that are part of a specific node. You can enable a specific server to have an SSL configuration associated with it so that resources within the same server can use the associated SSL configuration.
Clusters	Specifies the clusters that are part of the cell. When an SSL configuration is associated with a cluster, all servers within the cluster will use the same SSL configuration unless specified at a lower level in the topology.
Nodegroups	Specifies the node groups that are part of the cell. When an SSL configuration is associated with a node group, all nodes within that node group may use the same SSL configuration unless one is specified at a lower scope in the topology or the specific end point has chosen a direct alias reference.
Secure port and transport	Specifies an endpoint name to associate with an SSL configuration when more specific SSL settings are needed at this level. You could select an alias directly at the endpoint panel; however, when you use **Secure port and transport**, you can maintain more centralized control of the SSL configuration and make changes more easily.

Dynamic inbound and outbound endpoint SSL configurations collection:

Use this page to manage dynamic endpoint Secure Sockets Layer (SSL) configurations, which represent associations between Secure Socket Layer (SSL) configurations and their target protocol, host, and port.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Dynamic outbound endpoint SSL configurations**.

When an outbound connection is attempted, this association is checked ahead of the SSL configuration scope association. Based on the target protocol,host,port, the outbound SSL configuration used can be different from the default specified in the SSL scope configuration.

Table 151. Dynamic inbound and outbound endpoint SSL configurations buttons. This table lists the dynamic inbound and outbound endpoint SSL configuration buttons.

Button	Resulting action
New	Adds a new dynamic outbound selection criteria. The outbound connection selects an SSL configuration based upon connection information, including DNS host name and domain, port, and protocol type. When an outbound connection is being made, the dynamic outbound selection criteria are queried for a match, and if found the SSL configuration associated is used.
Delete	Deletes an existing dynamic outbound endpoint SSL configuration.

Name:

Specifies the unique name of the dynamic endpoint configuration.

Connection information:

Specifies the set of target protocol, host, port for the outbound request in the form *protocol*,*host*,*port*.

SSL Configuration:

Specifies the SSL configuration that is used by requests at this scope when a match occurs for the given selection criteria.

Dynamic outbound endpoint SSL configuration settings:

Use this page to set properties for dynamic outbound endpoint SSL configurations, which represent associations between SSL configurations and their target protocol, host, and port.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Dynamic [inbound | outbound] endpoint SSL configurations**. Then click the **New** button.

When an outbound connection is attempted, this association is checked ahead of the Secure Sockets Layer (SSL) configuration scope association. This means based on the target protocol,host,port, the outbound SSL configuration used can be different than the default specified in the SSL scope configuration.

Name:

Specifies the unique name of the dynamic endpoint configuration.

Data type: Text

Description:

Specifies text that describes the purpose of this dynamic selection criteria.

Data type: Text

Add connection information:

Specifies select information in the form protocol,host,port for the outbound connection. Multiple selection criteria can be entered. All of the connection information for dynamic outbound selection might not be available, and you may have to adjust the dynamic outbound selection connection filter and fill in an asterisk (*) for the missing part of the connection information. An asterisk (*) can be used to mean all protocols, hosts, or ports. You can use an asterisk(*) for any field.

Data type: Text

An example of selection criteria is `*,www.ibm.com,*`, which means that any time the target host is www.ibm.com, you must use the SSL configuration specified here. Another example selection criteria is `IIOP,*,*`, which means that any outbound IIOP request uses the SSL configuration that is specified in the SSL configuration field. When there is a conflict between two selection criteria, the application server uses the first match. The list of valid protocols you can use include: IIOP, HTTP, JMS, LDAP, SIP, ADMIN_SOAP, ADMIN_IIOP, or WEBSERVICES_HTTP.

When user written applications are expecting to take advantage of dynamic outbound selections, know that not all connection information may be available. For example, the openConnection() call on an URL object ultimately calls `createSocket(java.net.Socket socket, String host, int port, boolean autoClose)`. The connection information can be built with the host and port provided, but there is no protocol provided. In this case, a wild card, an asterisk (*), should be used for the protocol part of the dynamic selection connection information.

Add:

Specifies to add the selected information from the **Add select information** menu to the right-hand list.

Remove:

Specifies to remove the selection from the right-hand list.

SSL Configuration:

Specifies the SSL configuration to be used by requests at this scope when a match occurs for the given selection criteria.

Data type: Text

Get certificate alias:

When selected, the keystore within the selected SSL configuration is queried for a list of personal certificates from which to choose.

Certificate alias:

Specifies the certificate alias that is used as the identity for the connection.

If you select **None**, the Java Secure Sockets Extension (JSSE) key manager determines which certificate is used. If multiple certificates exist in the keystore, the key manager might not consistently select the same certificate.

Data type: Text
Default: (none)

Quality of protection (QoP) settings

Use this page to specify security level, ciphers, and mutual authentication settings for the Secure Socket Layer (SSL) configuration.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **SSL configurations >** . Click on {*SSL_configuration_name* }. Under **Additional Properties**, click **Quality of protection (QoP) settings**.

Client authentication:

Specifies the whether SSL client authentication should be requested if the SSL connection is used for the server side of the connection.

If `None` is selected, the server does not request that a client certificate be sent during the handshake. If `Supported` is selected, the server requests that a client certificate be sent. If the client does not have a certificate, the handshake might still succeed. If `Required` is selected, the server requests that a client certificate be sent. If the client does not have a certificate, the handshake fails.

Data type: Text
Default: None

Protocol:

Specifies the Secure Sockets Layer (SSL) handshake protocol. This protocol is typically SSL_TLS, which supports all handshake protocols except for SSLv2 on the server side. When United States Federal Information Processing standard (FIPS) option is enabled, Transport Layer Security (TLS) is automatically used regardless of this setting.

Data type: text
Default: SSL_TLS

Predefined JSSE provider:

Specifies one of the predefined Java Secure Sockets Extension (JSSE) providers. The IBMJSSE2 provider is recommended for use on all platforms which support it. It is required for use by the channel framework SSL channel. When Federal Information Processing Standard (FIPS) is enabled, IBMJSSE2 is used in combination with the IBMJCEFIPS crypto provider.

Default: Enabled

Select provider:

Specifies a package that implements a subset of the cryptography aspects for the Java security application programming interface (API). This value is a JSSE provider name that is listed in the java.security file. Note that cipher suites and protocol values depend upon the provider.

Data type: Text
Default: IBMJSSE2

Custom JSSE provider:

Specifies that a custom JSSE provider should be used.

Default: Disabled

Custom provider:

Specifies a package that implements a subset of the cryptography aspects for the Java security application programming interface (API). This value is a Java Secure Sockets Extension (JSSE) provider name that is listed in the java.security file. Note that cipher suites and protocol values depend upon the provider.

Data type: Text

Cipher suite groups:

Specifies the various cipher suite groups that can be chosen depending upon your security needs. The stronger the cipher suite strength, the better the security; however, this can result in performance consequences.

Data type: Text
Default: Strong

Update selected ciphers:

When selected, the cipher suites that are contained within the selected Cipher suite group are added to the list of **Selected ciphers**. Any change to this list changes the Cipher suite group to `custom`.

Selected ciphers:

Specifies the ciphers that are effective when the configuration is saved. These ciphers are used to negotiate with the remote side of the connection during the handshake. A common cipher needs to be selected or the handshake fails.

Data type: Text

Add:

Specifies to add the selected cipher to the **Selected ciphers** list.

Remove:

Specifies to remove the selected cipher from the **Selected ciphers** list.

ssl.client.props client configuration file

Use the `ssl.client.props` file to configure Secure Sockets Layer (SSL) for clients. In previous releases of WebSphere Application Server, SSL properties were specified in the `sas.client.props` or `soap.client.props` files or as system properties. By consolidating the configurations, WebSphere Application Server enables you to manage security in a manner that is comparable to server-side configuration management. You can configure the `ssl.client.props` file with multiple SSL configurations.

Setting up the SSL configuration for clients

Client runtimes are dependent on the WebSphere Application Server `ssl.client.props` configurations.

Use the `setupCmdLine` script on the command line to specify the com.ibm.SSL.ConfigURL system property. The setupclient script also sets the CLIENTSSL variable.The com.ibm.SSL.ConfigURL property references a file URL that points to the `ssl.client.props` file. You can reference the CLIENTSSL variable on the command line of any script that uses the `setupCmdLine` file.

When you specify the com.ibm.SSL.ConfigURL system property, the SSL configuration is available to all protocols that use SSL. SSL configurations, which are referenced in the `ssl.client.props` file, also have aliases that you can reference. In the following sample code from the `sas.client.props` file, all of the SSL properties are replaced with a property that points to an SSL configuration in the `ssl.client.props` file:

```
com.ibm.ssl.alias=DefaultSSLSettings
```

The following sample code shows a property in the `soap.client.props` file that is similar to the com.ibm.SSL.ConfigURL property. This property references a different SSL configuration on the client side:

```
com.ibm.ssl.alias=DefaultSSLSettings
```

In the `ssl.client.props` file, you can change the administrative SSL configuration to avoid modifying the `soap.client.props` file.

Tip: Support for SSL properties is still specified in the `sas.client.props` and `soap.client.props` files. However, consider moving the SSL configurations to the `ssl.client.props` file, because this file is the new configuration model for client SSL.

When you are configuring a client which does not call setupCmdLine.sh to connect to an application server using security, you must ensure that the following system property is defined on the client configuration:

```
-Djava.security.properties=profile_root/properties/java.security
```

Properties of the ssl.client.props file

This section describes the default `ssl.client.props` file properties in detail, by sections within the file. Be aware that if you specify javax.net.ssl system properties, these will override the settings in `ssl.client.props` file.

Global properties

Global SSL properties are process-specific properties that include Federal Information Processing Standard (FIPS) enablement, the default SSL alias, the profile_root of the profile for specifying the root location of the key and truststore paths, and so on.

Table 152. Properties of the ssl.client.props file. This table describes the properties of the `ssl.client.props` file.

Property	Default	Description
com.ibm.ssl.defaultAlias	DefaultSSLSettings	Specifies the default alias that is used whenever an alias is not specified by the protocol that calls the JSSEHelper API to retrieve an SSL configuration. This property is the final arbiter on the client side for determining which SSL configuration to use.
com.ibm.ssl.validationEnabled	false	When set to `true`, this property validates each SSL configuration as it is loaded. Use this property for debug purposes only, to avoid unnecessary performance overhead during production.
com.ibm.ssl.performURLHostNameVerification	false	When set to `true`, this property enforces URL host name verification. When HTTP URL connections are made to target servers, the common name (CN) from the server certificate must match the target host name. Without a match, the host name verifier rejects the connection. The default value of `false` omits this check. As a global property, it sets the default host name verifier. Any javax.net.ssl.HttpsURLConnection object can choose to enable host name verification for that specific instance by calling the setHostnameVerifier method with its own HostnameVerifier instance. **gotcha:** This property does not apply to SSL channels.
com.ibm.security.useFIPS	false	When set to `true`, FIPS-compliant algorithms are used for SSL and other Java Cryptography Extension (JCE)-specific applications. This property is typically not enabled unless the property is required by the operating environment.

Certificate creation properties

Use certificate creation properties to specify the default self-signed certificate values for the major attributes of a certificate. You can define the distinguished name (DN), expiration date, key size, and alias that are stored in the keystore.

Table 153. Certificate creation properties. This table describes the certificate creation properties.

Property	Default	Description
com.ibm.ssl.defaultCertReqAlias	default_alias	This property specifies the default alias to use to reference the self-signed certificate that is created in the keystore. If the alias already exists with that name, the default alias is appended with `_#`, where the number sign (#) is an integer that starts with 1 and increments until it finds a unique alias.
com.ibm.ssl.defaultCertReqSubjectDN	cn=${*hostname*}, o=IBM,c=US	This property uses the property distinguished name (DN) that you set for the certificate when it is created. The ${*hostname*} variable is expanded to the host name on which it resides. You can use correctly formed DNs as specified by the X.509 certificate.

Table 153. Certificate creation properties (continued). This table describes the certificate creation properties.

Property	Default	Description
com.ibm.ssl.defaultCertReqDays	365	This property specifies the validity period for the certificate and can be as small as 1 day and as large as the maximum number of days that a certificate can be set, which is approximately 15 years.
com.ibm.ssl.defaultCertReqKeySize	1024	This property is the default key size. The valid values depend upon the Java Virtual Machine (JVM) security policy files that are installed. By default, the product JVMs ship with the export policy file that limits the key size to 1024. To get a large key size such as 2048, you can download the restricted policy files from the website.

Certificate revocation checking

To enable certificate revocation checking, you can set a combination of Online Certificate Status Protocol (OCSP) properties. These properties are not used unless you set the com.ibm.ssl.trustManager property to `IbmPKIX`. In addition, to successfully process revocation checking on the client, you must turn off the signer exchange prompt. To turn off the signer exchange prompt, change the com.ibm.ssl.enableSignerExchangePrompt property to `false`. For more information, see the related link to the "Enabling certificate revocation checking with the default IbmPKIX trust manager" topic.

SSL configuration properties

Use the SSL configuration properties section to set multiple SSL configurations. For a new SSL configuration specification, set the com.ibm.ssl.alias property because the parser starts a new SSL configuration with this alias name. The SSL configuration is referenced by using the alias property from another file, such as `sas.client.props` or `soap.client.props`, through the default alias property. The properties that are specified in the following table enable you to create a javax.net.ssl.SSLContext, among other SSL objects.

Table 154. SSL configuration properties. This table lists the SSL configuration properties.

Property	Default	Description
com.ibm.ssl.alias	DefaultSSLSettings	This property is the name of this SSL configuration and must be the first property for an SSL configuration because it references the SSL configuration. If you change the name of this property after it is referenced elsewhere in the configuration, the runtime defaults to the com.ibm.ssl.defaultAlias property whenever the reference is not found. The error `trust file is null` or `key file is null` might display when you start an application using an SSL reference that is no longer valid.
com.ibm.ssl.protocol	SSL_TLS	This property is the SSL handshake protocol that is used for this SSL configuration. This property attempts Transport Layer Security (TLS) first, but accepts any remote handshake protocol, including SSLv3 and TLS. Valid values for this property include SSLv2 (client side only), SSLv3, SSL, TLS, TLSv1, and SSL_TLS.
com.ibm.ssl.securityLevel	STRONG	This property specifies the cipher group that is used for the SSL handshake. The typical selection is STRONG, which specifies 128-bit or higher ciphers. The MEDIUM selection provides 40-bit ciphers. The WEAK selection provides ciphers that do not perform encryption, but do perform signing for data integrity. If you specify your own cipher list selection, uncomment the property com.ibm.ssl.enabledCipherSuites. **Note:** The use of javax.net.ssl system properties causes this value to always be HIGH.

Table 154. SSL configuration properties (continued). This table lists the SSL configuration properties.

Property	Default	Description
com.ibm.ssl.trustManager	IbmX509	This property specifies the default trust manager that you must use to validate the certificate sent by the target server. This trust manager does not perform certificate revocation list (CRL) checking. You can choose to change this value to IbmPKIX for CRL checking using CRL distribution lists in the certificate, which is a standard way to perform CRL checking. When you want to perform custom CRL checking, you must implement a custom trust manager and specify the trust manager in the com.ibm.ssl.customTrustManagers property. The IbmPKIX option might affect performance because this option requires IBMCertPath for trust validation. Use IbmX509 unless CRL checking is necessary. If you are using the Online Certificate Status Protocol (OCSP) properties, set this property value to `IbmPKIX`.
com.ibm.ssl.keyManager	IbmX509	This property specifies the default key manager to use for choosing the client alias from the specified keystore. This key manager uses the com.ibm.ssl.keyStoreClientAlias property to specify the keystore alias. If this property is not specified, the choice is delegated to Java Secure Socket Extension (JSSE). JSSE typically chooses the first alias that it finds.
com.ibm.ssl.contextProvider	IBMJSSE2	This property is used to choose the JSSE provider for the SSL context creation. It is recommended that you default to IBMJSSE2 when you use a Java virtual machine (JVM). The client plug-in can use the SunJSSE provider when using a Sun JVM.
com.ibm.ssl.enableSignerExchangePrompt	true	This property determines whether to display the signer exchange prompt when a signer is not present in the client truststore. The prompt displays information about the remote certificate so that WebSphere Application Server can decide whether or not to trust the signer. It is very important to validate the certificate signature. This signature is the only reliable information that can guarantee that the certificate has not been modified from the original server certificate. For automated scenarios, disable this property to avoid SSL handshake exceptions. Run the `retrieveSigners` script, which sets up the SSL signer exchange, to download the signers from the server prior to running the client. If you are using the Online Certificate Status Protocol (OCSP) properties, set this property value to `false`.
com.ibm.ssl.keyStoreClientAlias	default	This property is used to reference an alias from the specified keystore when the target does not request client authentication. When WebSphere Application Server creates a self-signed certificate for the SSL configuration, this property determines the alias and overrides the global com.ibm.ssl.defaultCertReqAlias property.
com.ibm.ssl.customTrustManagers	Commented out by default	This property enables you to specify one or more custom trust managers, which are separated by commas. These trust managers can be in the form of *algorithm*|*provider* or *classname*. For example, IbmX509IIBMJSSE2 is in the *algorithm*|*provider* format, and the com.acme.myCustomTrustManager interface is in the *classname* format. The class must implement the javax.net.ssl.X509TrustManager interface. Optionally, the class can implement the com.ibm.wsspi.ssl.TrustManagerExtendedInfo interface. These trust managers run in addition to the default trust manager that is specified by the com.ibm.ssl.trustManager interface. These trust managers do not replace the default trust manager.

Table 154. SSL configuration properties (continued). This table lists the SSL configuration properties.

Property	Default	Description
com.ibm.ssl.customKeyManager	Commented out by default	This property enables you to have one, and only one, custom key manager. The key manager replaces the default key manager that is specified in the com.ibm.ssl.keyManager property. The form of the key manager is *algorithm*|*provider* or *classname*. See the format examples for the com.ibm.ssl.customTrustManagers property. The class must implement the javax.net.ssl.X509KeyManager interface. Optionally, the class can implement the com.ibm.wsspi.ssl.KeyManagerExtendedInfo interface. This key manager is responsible for alias selection.
com.ibm.ssl.dynamicSelectionInfo	Commented out by default	This property enables dynamic association with the SSL configuration. The syntax for a dynamic association is *outbound_protocol*, *target_host*, or *target_port*. For multiple specifications, use the vertical bar (|) as the delimiter. You can replace any of these values with an asterisk (*) to indicate a wildcard value. Valid *outbound_protocol* values include: IIOP, HTTP, LDAP, SIP, BUS_CLIENT, BUS_TO_WEBSPHERE_MQ, BUS_TO_BUS, and ADMIN_SOAP. When you want the dynamic selection criteria to choose the SSL configuration, uncomment the default property, and add the connection information. For example, add the following on one line `com.ibm.ssl.dynamicSelectionInfo=HTTP, .ibm.com,443|HTTP,.ibm.com,9443`
com.ibm.ssl.enabledCipherSuites	Commented out by default	This property enables you to specify a custom cipher suite list and override the group selection in the com.ibm.ssl.securityLevel property. The valid list of ciphers varies according to the provider and JVM policy files that are applied. For cipher suites, use a space as the delimiter.
com.ibm.ssl.keyStoreName	ClientDefaultKeyStore	This property references a keystore configuration name. If you have not already defined the keystore, the rest of the keystore properties must follow this property. After you define the keystore, you can specify this property to reference the previously specified keystore configuration. New keystore configurations in the `ssl.client.props` file have a unique name.
com.ibm.ssl.trustStoreName	ClientDefaultTrustStore	This property references a truststore configuration name. If you have not already defined the truststore, the rest of the truststore properties must follow this property. After you define the truststore, you can specify this property to reference the previously specified truststore configuration. New truststore configurations in the `ssl.client.props` file should have a unique name.

Keystore configurations

SSL configurations reference keystore configurations whose purpose is to identify the location of certificates. Certificates represent the identity of clients that use the SSL configuration. You can specify keystore configurations with other SSL configuration properties. However, it is recommended that you specify the keystore configurations in this section of the `ssl.client.props` file after the com.ibm.ssl.keyStoreName property identifies the start of a new keystore configuration. After you fully define the keystore configuration, the com.ibm.ssl.keyStoreName property can reference the keystore configuration at any other point in the file.

Table 155. Keystore configuration properties. This table lists the keystore configuration properties.

Property	Default	Description
com.ibm.ssl.keyStoreName	ClientDefaultKeyStore	This property specifies the name of the keystore as it is referenced by the runtime. Other SSL configurations can reference this name further down in the `ssl.client.props` file to avoid duplication.

Table 155. Keystore configuration properties (continued). This table lists the keystore configuration properties.

Property	Default	Description
com.ibm.ssl.keyStore	${user.root}/etc/key.p12	This property specifies the location of the keystore in the required format of the com.ibm.ssl.keyStoreType property. Typically, this property references a keystore file name. However, for cryptographic token types, this property references a Dynamic Link Library (DLL) file. **gotcha:** If you are using a 4764 cryptography card, then the keystore file name for the client configuration should be specified as the file `4764.cfg` in a directory structure of your choice, and the corresponding com.ibm.ssl.keyStoreType should be set to PKCS11. The 4764.cfg file is NOT supplied with WebSphere Application Server.
com.ibm.ssl.keyStorePassword	WebAS	This property is the default password, which is the cell name for the profile when it is created. The password is typically encoded using an {xor} algorithm. You can use iKeyman to change the password in the keystore, then change this reference. If you do not know the password and if the certificate is created for you, change the password in this property, then delete the keystore from the location where it resides. Restart the client to recreate the keystore by using the new password, but only if the keystore name ends with DefaultKeyStore and if the fileBased property is `true`. Delete both the keystore and truststore at the same time so that a proper signer exchange can occur when both are recreated together.
com.ibm.ssl.keyStoreType	PKCS12	This property is the keystore type. Use the default, PKCS12, because of its interoperability with other applications. You can specify this property as any valid keystore type that is supported by the JVM on the provider list.
com.ibm.ssl.keyStoreProvider	IBMJCE	The IBM Java Cryptography Extension property is the keystore provider for the keystore type. The provider is typically IBMJCE or IBMPKCS11Impl for cryptographic devices.
com.ibm.ssl.keyStoreFileBased	true	This property indicates to the runtime that the keystore is file-based, meaning it is located on the file system.
com.ibm.ssl.keyStoreReadOnly	false	This property indicates to the run time for WebSphere Application Server whether the key store can be modified during the run time.

Truststore Configurations

SSL configurations reference truststore configurations, whose purpose is to contain the signer certificates for servers that are trusted by this client. You can specify these properties with other SSL configuration properties. However, it is recommended that you specify truststore configurations in this section of the `ssl.client.props` file after the com.ibm.ssl.trustStoreName property has identified the start of a new truststore configuration. After you fully define the truststore configuration, the com.ibm.ssl.trustStoreName property can reference the configuration at any other point in the file.

A truststore is a keystore that JSSE uses for trust evaluation. A truststore contains the signers that WebSphere Application Server requires before it can trust the remote connection during the handshake. If you configure the com.ibm.ssl.trustStoreName=ClientDefaultKeyStore property, you can reference the

keystore as the truststore. Further configuration is not required for the truststore because all of the signers that are generated through signer exchanges are imported into the keystore where they are called by the runtime.

Table 156. Truststore Configuration properties. This table lists the truststore configuration properties.

Property	Default	Description
com.ibm.ssl.trustStoreName	ClientDefaultTrustStore	This property specifies the name of the truststore as it is referenced by the runtime. Other SSL configurations can reference further down in the `ssl.client.props` file to avoid duplication.
com.ibm.ssl.trustStore	${user.root}/etc/trust.p12	This property specifies the location of the truststore in the format that is required by the truststore type that is referenced by the com.ibm.ssl.trustStoreType property. Typically, this property references a truststore file name. However, for cryptographic token types, this property references a DLL file. **gotcha:** If you are using a 4764 cryptography card, then the keystore file name for the client configuration should be specified as the file `4764.cfg` in a directory structure of your choice, and the corresponding com.ibm.ssl.keyStoreType should be set to PKCS11. The 4764.cfg file is NOT supplied with WebSphere Application Server.
com.ibm.ssl.trustStorePassword	WebAS	This property specifies the default password, which is the cell name for the profile when it is created. The password is typically encoded using an {xor} algorithm. You can use iKeyman to change the password in the keystore, then change the reference in this property. If you do not know the password and if the certificate was created for you, change the password in this property, then delete the truststore from the location where it resides. Restart the client to recreate the truststore by using the new password, but only if the keystore name ends with DefaultTrustStore and the fileBased property is `true`. It is recommended that you delete the keystore and the truststore at the same time so that a proper signer exchange can occur when both are recreated together.

Table 156. Truststore Configuration properties (continued). This table lists the truststore configuration properties.

Property	Default	Description
com.ibm.ssl.trustStoreType	PKCS12	This property is the truststore type. Use the default PKCS12 type because of its interoperability with other applications. You can specify this property as any valid truststore type that is supported by the JVM functionality on the provider list.
com.ibm.ssl.trustStoreProvider	IBMJCE	This property is the truststore provider for the truststore type. The provider is typically IBMJCE or IBMPKCS11Impl for cryptographic devices.
com.ibm.ssl.trustStoreFileBased	true	This property indicates to the runtime that the truststore is file-based, meaning it is located on the file system.
com.ibm.ssl.trustStoreReadOnly	false	This property indicates to the run time for WebSphere Application Server whether the truststore can be modified during the run time.

Creating a CA client in SSL

A plug point is provided to allow users to connect to a certificate authority (CA) to request, query, and revoke certificates. A security configuration object, called a CAClient, must be created for WebSphere to communicate with the CA. The CAClient object must contain a WSPKIClient() implementation, and it will handle the connection and communicate with the CA server. Users can also create there own implementation.

Before you begin

The WSPKIClient interface must be implemented and the class name provided as part of the CAClient when it is created.

You use the administrative console to create a new CA client.

Procedure

1. Click **Security > SSL certificate and key management**.
2. Click **Certificate Authority (CA) client configurations**. A panel of existing CA clients appears.
3. Click **New** to create a new CA client in the configuration.

   Note: You can also create a CA client by using the **createCAClient** AdminTask .
4. Fill in the following information for the CA client
   - Name of the CA client.
   - The management scope (selected from the drop-down list).
   - WSPKIClient implementation class.
   - CA server host name.
   - User name.
   - Password.
   - Confirm of password.
   - Number of times to poll.

- Polling interval (in minutes) when requestin certificates.
- Custom properties.

5. Click **Apply** then **OK**.

Results

The information in the object can then be used by the runtime to connect to a CA to create, revoke, or replace a certificate.

Deleting a CA client in SSL

You can delete the CAClient object from the security configuration if a connection to a certificate authority (CA) is no longer needed.

Before you begin

You use the administrative console to delete a CA client.

Procedure

1. Click **Security > SSL certificate and key management**.
2. Click **Certificate Authority (CA) client configurations**. A panel displaying the existing CA clients appears.
3. Click the CA client name you want to delete.
4. Click the **Delete** button.

Note: You can also use the **deleteAClient** AdminTask to delete the CA client.

Results

The CA client is deleted from the configuration.

Note: When you use the deleteCAClient AdminTask to delete the CA client, the CA client cannot be deleted if a CA certificate that exists in the keystore was obtained from the certificate authority and is still referenced by the CA client. For example, when such CA certificate still exists, the user receives the following message:

```
wsadmin>$AdminTask deleteCAClient {-caClientName myca}
WASX7015E: Exception running command:
 "$AdminTask deleteCAClient {-caClientName myca}"; exception information:
 com.ibm.websphere.management.cmdframework.CommandValidationException:
 CWPKI0687E: The Certificate Authority (CA) client myca is still referenced by:
 [Certificate alias myca21 in key store CellDefaultKeyStore].
wsadmin>
```

Viewing or modifying a CA client in SSL

You can view or modify the CAClient object settings in the security configuration. The CAClient object contains all the information needed to connect and communicate with a certificate authority (CA). A connection to a Certificate Authority is used to request a certificate, query a certificate, or revoke a certificate.

Before you begin

You use the administrative console to view or modify a CA client.

Procedure

1. Click **Security > SSL certificate and key management**.

2. Click **Certificate Authority (CA) client configurations**. A panel displaying the existing CA clients appears.
3. Click the CA client name you want to examine and modify.

   Note: You can also use the **getCAClient** AdminTask to get information about the existing CA client and the **modifyCAClieint** AdminTask to make changes to the CA client.
4. Make the changes to the CA client information as required. Modify the following information as required.
   - Name of the CA client.
   - The management scope (selected from the drop-down list.
   - Implementation class.
   - CA server host name.
   - User name.
   - Password.
   - Confirm of password.
   - Number of times to poll.
   - Polling interval (in minutes) when requestin certificates.
   - Custom properties.
5. Click **Apply** then **OK**.

Results

The information in the object can then be used by the runtime to connect to a CA to create, revoke, or replace a certificate

What to do next

Creating a keystore configuration for a preexisting keystore file

A Secure Sockets Layer (SSL) configuration references keystore configurations during WebSphere Application Server runtime. Whether a keystore file was created by another keystore tool or saved from a previous configuration, the file must be referenced by a keystore configuration object to be used by the server. A keystore configuration object can be created to reference a pre-existing keystore object.

Before you begin

A keystore must already exist.

Alternative Method: To create a keystore by using the wsadmin tool, use the **createKeyStore** command of the AdminTask object. For more information, see the KeyStoreCommands command group for the AdminTask object article.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound}**.
2. Under Related Items, click **Key stores and certificates**, then click **New**.
3. Type a name in the **Name** field. This name uniquely identifies the keystore in the configuration.

4. Type the location of the keystore file in the **Path** field. The location can be a file name or a file URL to an existing keystore file.
5. Type the keystore password in the **Password** field. This password is for the keystore file that you specified in the **Path** field.
6. Type the keystore password again in the **Confirm Password** field to confirm the password.
7. Select a keystore type from the list. The type that you select is for the keystore file that you specified in the **Path** field.
8. Select any of the following optional selections:
   - The **Read only selection** creates a keystore configuration object but does not create a keystore file. If this option is selected, the keystore file that you specified in the **Path** field must already exist.
   - The **Initialize at startup selection** initializes the keystore during runtime.
   - The **Enable cryptographic operations on a hardware device** specifies whether a hardware cryptographic device is used for cryptographic operations only.

     Note: Operations that require login are not supported when using this option.
9. Click **Apply** and **Save**.

Results

You have created a keystore configuration object for the keystore file that you specified. This keystore can now be used in an SSL configuration.

What to do next

You can create additional keystore configurations, as needed.

Recreating the .kdb keystore internal password record

The IBM i keystore type `IBMi5OSKeyStore` does not recognize or generate `.sth` password stash files. Instead it keeps an internal record of the password for the `.kdb` keystore file where it is created. If the `.kdb` file is moved, the password is no longer associated with the keystore. In that case, you must use the Digital Certificate Manager (DCM) to recreate the internal record of the password for the `.kdb` keystore file.

Before you begin

Refer to the topic “Keystore configurations for SSL” on page 1715 before attempting this task.

About this task

To recreate the internal record of the password for the `.kdb` keystore file, start the DCM. For more information, see the Digital Certificate Manager information.

Procedure

1. Click **Select a Certificate Store**.
2. Select **Other System Certificate Store**.
3. Enter the certificate store path and filename.
4. Enter the certificate store password.
5. Click **Continue**.
6. In the left hand panel, select **Manage Certificate Store**.
7. Click **Change password**.
8. Enter the new password and confirm it. Note that DCM requires a different password than the one you specified in step 4.
9. Select **Automatic login**.

10. Click **Continue**.
11. Click **OK** when a message displays that confirms that the password is changed.
12. Repeat steps 1 through 5 to create the internal record of the new password for the `.kdb` keystore file.
13. Repeat steps 1 through 12 to change the password back to the original password and to create the internal record of the original password for the `.kdb` keystore file.

Results

You have recreated the internal record of the password for the `.kdb` keystore file.

Configuring a hardware cryptographic keystore

You can create a hardware cryptographic keystore that WebSphere Application Server can use to provide cryptographic token support in the server configuration.

About this task

Note: The hardware accelerator is not supported except for the following situations:

- If you are using WebSphere Application Server for z/OS and are using the IBMJCECCA crypto provider.
- If you are using WebSphere Application Server Version 7.0 and above running on zLinux and are using the IBMPKCS11 provider.

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Key stores and certificates**.
2. Click **New**.
3. Type a name to identify the keystore. This name is used to enable hardware cryptography in the Web Services Security configuration.
4. Optionally, you can type a description for the keystore in the **Description** field.
5. You can specify a **Management scope** for the key store. This is not required. The management scope specifies the scope where this Secure Sockets Layer (SSL) configuration is visible. For example, if you choose a specific node, then the configuration is only visible on that node and any servers that are part of that node.
6. Type the path for the hardware device-specific configuration file. The configuration file is a text file that contains entries in the following format: *attribute* = *value*. The valid values for attribute and value are described in detail in the Software Developer Kit, Java Technology Edition documentation. The two mandatory attributes are name and library, as shown in the following sample code:

   ```
   name = FooAccelerator
   library = /opt/foo/lib/libpkcs11.so
   slotListIndex = 0
   ```

   The configuration file should also include device-specific configuration data. Navigate to the PKCS11ImplConfigSamples.jar file, which contains sample configuration files, under the heading "PKCS 11 Implementation Provider" on the Java technology site http://www.ibm.com/developerworks/java/jdk/security/60/.

   Note: JSSE2 is unable to use the IBMPKCS11Impl provider for acceleration.

   a. You can use this link http://www.ibm.com/developerworks/java/jdk/security/50/secguides/pkcs11implDocs/IBMJavaPKCS11ImplementationProvider.html to initialize the IBMPKCS11 provider in a thread safe way

b. Specify a unique .cfg file that contains information about the supported hardware device. A list of supported hardware devices are available at http://www.ibm.com/developerworks/java/jdk/security/50/secguides/pkcs11implDocs/IBMPKCS11SupportList.html

c. You specify the Signature.getInstance method with the properly initialized IBMPKCS11Impl provider instance as shown.

```
Signature.getInstance("SHA1withRSA", ibmpkcs11implinstance);
```

7. Type a password if the token login is required. Operations that use keys on the token require a secure login. This field is optional if the keystore is used as a cryptographic accelerator. In this case, you need to select **Enable cryptographic operations on hardware device**.
8. Select the **PKCS11** type.
9. Select **Read only**.
10. Click **OK** and **Save**.

Results

WebSphere Application Server can now provide cryptographic token support in the server configuration.

Managing keystore configurations remotely

You can manage keystores remotely in a WebSphere Application Server, Network Deployment environment on separate machines. A node server can hold the configuration for a keystore, while the actual keystore resides on another system. After you set up a remotely managed configuration, you can perform all of the certificate and keystore operations for the keystore on the remote machine from the server that contains the keystore remote configuration.

Before you begin

Key stores can be remotely managed only in network deployed environments.

Alternative Method: To manage a self-signed certificates by using the wsadmin tool, use the **PersonalCertificateCommands** group commands of the AdminTask object. For more information, see the PersonalCertificateCommands command group for the AdminTask object article.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key stores and certificates**.
2. Click **New**.
3. Type a name in the **Name** field. This name uniquely identifies the keystore in the configuration.
4. Type the location of the keystore file in the **Path** field. The location can be a file name or a file Uniform Resource Locator (URL) to an existing keystore file.
5. Type the keystore password in the **Password** field. This password is for the keystore file that you specified in the **Path** field.
6. Type the keystore password again in the **Confirm Password** field to confirm the password.
7. Select a keystore type from the list. The type you select is for the keystore file that you specified in the **Path** field.
8. Select the **Remotely managed** check box, and then fill in one or more hosts names of the systems where the keystore file is to be located. If you provide multiple host names, separate the host names with a pipe (**|**).
9. Select any of the following optional selections:

- The **Read only selection** creates a keystore configuration object but does not create a keystore file. If this option is selected, the keystore file that you specified in the **Path** field must already exist.
- The **Initialize at startup selection** initializes the keystore during run time.

10. Select **Apply** and **Save**.

Results

A keystore configuration object is created on the server from where the command was run. The keystore file for the configuration will be created on each system that you specified in the host list.

What to do next

Now, you can perform all certificate management operations on the keystore from the system where the keystore configuration resides. For example, you can perform certificate management operations, such as: creating a self-signed certificate, extracting a certificate, or extracting a signer certificate.

Keystores and certificates collection

Use this page to manage keystore types, including cryptography, Resource Access Control Facility (RACF), Certificate Management Services (CMS), Java, and all trust store types.

gotcha: In most cases, having unused and expired signer certificates in a trust store does not cause problems. However, if you experience a problem because the trust store includes an unused or expired signer certificate, you can safely delete the following expired signer certificates from the dummy keystores files:

- DummyClientKeyFile.jks
- DummyClientTrustFile.jks
- DummyServerKeyFile.jks
- DummyServerTrustFile.jks

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Keystores and certificates**.

Table 157. Keystores and certificates buttons. This table describes the keystores and certificates buttons.

Button	Resulting action
New	Adds a new keystore object that can be referenced by Secure Sockets Layer (SSL) configurations or KeySets. The Keystore management scope is based on the part of the topology tree from which it was created.
Delete	Deletes an existing keystore. The keystore should not be referenced by any other parts of the configuration before you delete it.
Change password	Allows for changing a keystore password.
Exchange signers	Refers to exchanging signers in a keystore. You can select two keystores, along with personal certificates or signer certificates from a selected keystore, then add them as a signer to another selected keystore.

Keystore usages: Filters the keystore usage types in the keystore collection.

The default value for the keystore usage filter depends on the navigation path that you followed to get to the Keystores and certificates panel. You can change the value of the keystore usage filter by clicking on the drop-down list and selecting a different filter value.

Navigation path	Keystore usage default value
Security > SSL certificate and key management > Keystores and certificates	SSL keystores

Navigation path	Keystore usage default value
Security > SSL certificate and key management > Key sets > CellLTPAKeyPair > Keystores and certificates	Key set keystores
Security > SSL certificate and key management > SSL configurations > CellDefaultSSLSettings > Keystores and certificates	SSL keystores
Security > SSL certificate and key management > Manage endpoint security configurations > *node name* > Keystores and certificates	SSL keystores

Name:

Specifies the unique name that is used to identify the keystore. This name is typically scoped by the ManagementScope scopeName and based upon the location of the keystore. The name must be unique within the existing keystore collection.

This is a user-defined name.

Description:

Specifies the description of the keystore.

This is a user-defined description.

Path:

Specifies the location of the keystore file in the format needed by the keystore type. This file can be a card-specific configuration file for cryptographic devices or a filename or file URL for file-based keystores. It can be a safkeyring URL for RACF keyrings.

Key store settings

Use this page to create all keystore types, including cryptographic, Resource Access Control Facility (RACF), Certificate Management Services (CMS), Java, and all truststore types.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound}**. Under Related Items, click **Key stores and certificates**. Click either **New** or an existing keystore.

Links to Personal certificates, Signer certificates, and Personal certificate requests enable you to manage certificates in a manner similar to iKeyman capabilities. A keystore can be file-based, such as CMS or Java keystore types, or it can be remotely managed.

Note: Any changes made to this panel are permanent.

Name:

Specifies the unique name to identify the keystore. The keystore is typically scoped by the ManagementScope scopeName based on the location of the keystore. The name must be unique within the existing keystore collection.

Data type: Text

Description:

Specifies the description of the keystore.

Data type: Text

Management scope:

Specifies the scope where this Secure Sockets Layer (SSL) configuration is visible. For example, if you choose a specific node, then the configuration is only visible on that node and any servers that are part of that node.

Data type: Text

Path:

Specifies the location of the keystore file in the format needed by the keystore type. This file can be a dynamic link library (DLL) for cryptographic devices or a filename or file URL for file-based keystores. It can be a safkeyring URL for RACF keyrings.

Data type: Text

Control region user:

Specifies the Control region Started Task user ID in which the Control region System Authorization Facility (SAF) keyring is created. The user ID must match the exact ID being used by the Control region. Note: This option only applies when creating writable SAF keyrings on z/OS.

Data type: Text

Servant region user:

Specifies the Servant region Started Task user ID in which the Servant region System Authorization Facility (SAF) keyring is created. The user ID must match the exact ID being used by the Servant region. Note: This option only applies when creating writable SAF keyrings on z/OS.

Data type: Text

Password [new keystore] / Password [existing keystore]:

Specifies the password used to protect the physical keystore in the operating system. For the default keystore (names ending in DefaultKeyStore or DefaultTrustStore), the password is `WebAS`. This default password must be changed.

This field can be edited.

Data type: Text

Note: If you want to push the key store to all nodes, the path should be: `${CONFIG_ROOT}/cells/CELLNAME/yourkeystore.kdb`.

Confirm password:

Specifies confirmation of the password to open the keystore file or device.

Data type: Text

Type:

Specifies the implementation for keystore management. This value defines the tool that operates on this keystore type.

The list of options is returned by java.security.Security.getAlgorithms("KeyStore"). Some options might be filtered and some might be added based on the java.security configuration.

Data type: Text
Default: PKCS12

Read only:

Specifies whether the keystore can be written to or not. If the keystore cannot be written to, certain operations cannot be performed, such as creating or importing certificates.

Default: Disabled

Remotely managed:

Specifies whether the key store is remotely managed, which means that a remote MBean call is needed to update the key store based on the host name specified in the host list field. Most hardware cryptographic token devices are remotely managed. If a key store is marked remotely managed, list the host name of the server where the device is installed in the Host list field.

Default:

Initialize at startup:

Specifies whether the keystore needs to be initialized before it can be used for cryptographic operations. If enabled, the keystore is initialized at server startup.

Default: Disabled

Enable cryptographic operations on hardware device:

Specifies whether a hardware cryptographic device is used for cryptographic operations only. Operations that require a login are not supported when using this option.

Default: Disabled

Key managers collection

Use this page to define the implementation settings for key managers. A key manager is invoked during a Secure Sockets Layer (SSL) handshake to determine which certificate alias is used. The default key manager (WSX509KeyManager) performs alias selection. If more advanced function is desired, define a custom key manager on the **Manage endpoint security configurations** panel.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key managers**.

Table 158. Key managers buttons. This table describes the key managers buttons.

Button	Resulting action
New	Adds a new key manager that can be selected by an SSL configuration. A key manager is invoked during an SSL handshake to select a specific certificate alias to use from a key store.
Delete	Deletes an existing key manager. The key manager should not be referenced by any SSL configuration before you can delete it.

Name:

Specifies the name of the key manager, which you can select on the SSL configuration panel.

Class name:

Specifies the name of the key manager implementation class. This class implements javax.net.ssl.X509KeyManager interface and, optionally, the com.ibm.wsspi.ssl.KeyManagerExtendedInfo interface.

Algorithm:

Specifies the algorithm name of the key manager that is implemented by the selected provider.

Key managers settings

Use this page to define key managers implementation settings. A key manager gets invoked during an Secure Sockets Layer (SSL) handshake to determine the certificate alias to be used. The default key manager (WSX509KeyManager) performs alias selection. If more advanced function is desired, a custom key manager can be specified here and selected in the SSL configuration.

To view this administrative console page, click **Security** > **SSL certificate and key management**. Under Configuration settings, click**Manage endpoint security configurations** > **{Inbound | Outbound}** > ***ssl_configuration***. Under Related items, click **Key managers**. On the next panel, click **New**.

Name:

Specifies the name of the key manager, which you can select on the SSL configuration panel.

Data type: Text

Management scope:

Specifies the scope where this Secure Sockets Layer (SSL) configuration is visible. For example, if you choose a specific node, then the configuration is only visible on that node and any servers that are part of that node.

Data type List
Range: Applicable scopes

Standard:

Specifies the key manager selection that is available from a Java provider that is installed in the java.security file. This provider might be shipped by Java Secure Sockets Extension (JSSE) or be a custom provider that implements an X509KeyManager interface.

Default: Enabled

Provider:

Specifies the provider name that has an implementation of an X509KeyManager interface. This provider is typically set to IBMJSSE2.

Data type: Text
Default: IBMJCE

Algorithm:

Specifies the algorithm name of the trust manager implemented by the selected provider.

Data type: Text
Default: IbmX509

Custom:

Specifies that the key manager selection is based on a custom implementation class that implements the javax.net.ssl.X509KeyManager interface and optionally the com.ibm.wsspi.ssl.KeyManagerExtendedInfo interface to obtain additional connection information not otherwise available.

Default: Disabled

Class name:

Specifies the name of the key manager implementation class.

Data type: Text

Creating a self-signed certificate

You can create a self-signed certificate. WebSphere Application Server uses the certificate at runtime during the handshake protocol. Self-signed certificates are located in the default keystore.

Before you begin

You must create a keystore before you can create a self-signed certificate.

Alternative Method: To create a self-signed certificate by using the wsadmin tool, use the **createSelfSignedCertificate** command of the AdminTask object. For more information, see the PersonalCertificateCommands command group for the AdminTask object article.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key stores and certificates > [*keystore*]**.
2. From Additional Properties, click **Personal certificates**.
3. Click **Create a self-signed certificate**.
4. Type a certificate alias name. The alias identifies the certificate request in the keystore.

5. Type a common name (CN) value. This value is the CN value in the certificate distinguished name (DN).
6. Type the validity period The default validity period value is 365 days.
7. You can configure one or more of the following optional values:
   a. Optional: Select a key size value. The default key size value is 2048 bits.
   b. Optional: Type an organization value. This value is the O value in the certificate DN.
   c. Optional: Type an organizational unit value. This organizational unit value is the OU value in the certificate DN.
   d. Optional: Type a locality value. This locality value is the L value in the certificate DN.
   e. Optional: Type a state or providence value. This value is the ST value in the certificate DN.
   f. Optional: Type a zip code value. This zip code value is the POSTALCODE value in the certificate DN.
   g. Optional: Select a country value from the list. This country value is the C= value in the certificate request DN.
8. Click **Apply**.

Results

You have created a self-signed certificate that resides in the keystore. The SSL configuration for the WebSphere Application Server runtime uses this certificate for SSL communication. Extract the signer of the self-signed certificate to add the signer to another keystore.

Replacing an existing personal certificate

Occasionally, you need to replace an existing personal certificate with a new certificate. This task discusses how to replace the existing personal certificate in the keystore. It searches all keystores for a signer certificate extracted from the original personal certificate, and places the signer of the new personal certificate in it's place. It also updates all of the certificate alias references in the security configuration with the new one.

Before you begin

The current certificate and the certificate replacement must exist in the same keystore before you can replace a certificate.

Alternative Method: To replace a self-signed certificate by using the wsadmin tool, use the **replaceCertificate** command of the AdminTask object. For more information, see the PersonalCertificateCommands command group for the AdminTask object article

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key stores and certificates > [*keystore*]**.
2. Under Additional Properties, click **Personal certificates**.
3. Select the certificate to be replaced. The alias list must include the certificate to be replaced and the certificate to replace it with.
4. Click **Replace**.
5. Select a replacement certificate alias from the list.
6. You can delete one of the following types of certificates:
   - Select **Delete old certificate** to delete the existing or expired certificate.

- Select **Delete old signers** to delete the existing signer certificates.

7. Click **Apply**.

Results

Your results depend on what you selected:

- If you selected **Delete old certificate**, the new certificate alias replaces all of the references to the certificate alias in the configuration.
- If you selected **Delete old signers**, the new signer certificate replaces all of the occurrences of the old signer certificates.
- If the new certificate alias replaces the existing alias, the WebSphere Application Server runtime checks to make sure that:
  - All of the SSL Configurations objects reference the certificate
  - The Dynamic SSL Configuration Selections objects and the SSL Configuration group objects reference the certificate.
- If you selected **Delete old signers**, the existing signer certificates are replaced.
- If you selected **Delete old certificate**, the existing certificate is deleted.

Creating a new SSL certificate to replace an existing one in a node

When using the -asExistingNode option on the addNode command, you might be adding an existing node to a different machine. The default Secure Sockets Layer (SSL) certificate of the node does not contain the name of the machine the node is located on. In most scenarios, the subject DN of the default certificate does not make a difference. However, you might want to change the default certificate of the node to contain the hostname of the node.

Before you begin

To replace the default certificate of a node, you must create a new NodeDefaultKeyStore for the certificate and then replace the old certificate with the new one.

The certificate created by default on the WebSphere Application Server subjectDN is of the form cn=<hostname>, ou=<cell name>, ou=<node name>, o=ibm, c=us. When creating a new certificate you can also customize the subjectDN.

About this task

To create a new SSL certificate in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Key stores and certificates**.
2. Select the NodeDefaultKeyStore of the node you want to change.
3. Under Additional Properties, select **Personal certificates**.
4. Under the Create pull-down, select **Chained Certificate**.
5. Enter a certificate and alias name. This can be any name you choose as long as the alias does not already exist. It is just a label to identify the certificate in the keystore.
6. Enter a common name. This is typically the hostname the node is running on.
7. Optional: Fill in any of the other Subject DN related fields. If you want the subject DN to look like the default subjectDN on WebSphere Application Server, then enter:
   - `IBM` in the Organization field.
   - `<cell name>,ou=<node name>` in the Organization unit field.
   - Under the Country or region pull-down, select **US**.

8. You can use the defaults for Root certificate used to sign the certificate, Key Size, and Validity Period or supply your own values.
9. Click **Apply**.

   Note: You can also create a new chained certificate using the createChainedCertificate command. Read PersonalCertificateCommands command group for the AdminTask object for more information.

   You must now replace the old certificate with the one you just created. The replace certificate option not only replaces the old default certificate with a new one but also replaces any occurrences of the signer of the old certificate with the signer of the new certificate. The configuration is also checked for references to the alias name of the old certificate and replaces it with the alias name of the new certificate. To replace the old certificate with the new one, complete the remaining steps.
10. Click **Security > SSL certificate and key management > Key stores and certificates**.
11. Select the NodeDefaultKeyStore of the node you want to change.
12. Under Additional Properties, select **Personal certificates**.
13. Select the default certificate of the node, usually called `default`.
14. Click **Replace**.
15. Select the certificate alias name for the certificate you just created from the **Replace with** pull-down.
16. Click **Delete old Certificate after replacement**.
17. Click **Apply**.

    Note: You can also create a new chained certificate using the replaceCertificate command. Read PersonalCertificateCommands command group for the AdminTask object for more information.

What to do next

You can also replace default certificates in an entire cell. Read Creating new SSL certificates to replace existing ones in a cell for more information.

Creating new SSL certificates to replace existing ones in a cell

To replace default Secure Socket Layer (SSL) certificates in an entire cell, you must create a new self-signed root certificate in the root keystore, DmgrDefaultRootStore, and replace the old root certificate with the new one.

About this task

For the default certificate of the cell in CellDefaultKeyStore and the default certificate of each node in NodeDefaultKeyStore, create a new chained certificate and replace the old default certificate with the new certificate.

The root certificate is created by default on WebSphere Application Server, and has a subjectDN in the form cn=<hostname>, ou=Root Certificate, ou=<cell name>, ou=<node name>, o=ibm, c=us. When you create a new root certificate you can also customize the subject DN.

To create a new SSL root certificate in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Key stores and certificates**.
2. Under the Keystore usages pull-down, select **Root certificate keystore**.
3. Select the DmgrDefaultRootStore in the keystore collection.
4. Under Additional Properties, select **Personal certificates**.
5. Under the Create pull-down, select **Self-signed Certificate**.

6. Enter a certificate and alias name. This can be any name you choose as long as the alias does not already exist. It is just a label to identify the certificate in the keystore.
7. Enter a common name. This is typically the hostname the node is running on.
8. Optional: Fill in any of the other Subject DN related fields. If you want the subject DN to look like the default subjectDN on WebSphere Application Server, then enter:
   - `IBM` in the Organization field.
   - `<cell name>,ou=<node name>` in the Organization unit field.
   - Under the Country or region pull-down, select **US**.
9. You can use the defaults for Root certificate used to sign the certificate, Key Size, and Validity Period or supply your own values.
10. Click **Apply**.

    Note: You can also create a self-signed certificate using the createSelfSignedCertificate command. Read PersonalCertificateCommands command group for the AdminTask object for more information.

    You must now replace the old root certificate with the one you just created. The replace certificate option not only replaces the old default certificate with a new one but also replaces any occurrences of the signer of the old certificate with the signer of the new certificate. The configuration is also checked for references to the alias name of the old certificate and replaces it with the alias name of the new certificate. To replace the old certificate with the new one, complete the remaining steps.
11. Click **Security > SSL certificate and key management > Key stores and certificates**.
12. Select the CellDefaultKeyStore of the node you want to change.
13. Under Additional Properties, select **Personal certificates**.
14. Select the default certificate of the node, usually called `default`.
15. Click **Replace**.
16. Select the certificate alias name for the new certificate you just created from the **Replace with** pull-down.
17. Click **Delete old Certificate after replacement**.
18. Click **Apply**.

What to do next

You can also replace default certificates in a node. Read Creating a new SSL certificate to replace an existing one in a node for more information

Creating a certificate authority request

To ensure Secure Sockets Layer (SSL) communication, servers require a personal certificate that is either self-signed, chained or signed by an external certificate authority (CA). You must first create a personal certificate request to obtain a certificate that is signed by a CA.

Before you begin

The keystore that contains a personal certificate request must already exist.

Alternative Method: To create a certificate request by using the wsadmin tool, use the **createCertificateRequest** command of the AdminTask object. For more information, see the CertificateRequestCommands command group of the AdminTask object article.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Key stores and certificates > *keystore***.
2. Click **Personal certificate requests > New**.
3. Type the full path of the certificate request file. The certificate request is created in this location.
4. Type an alias name in the **Key label** field. The alias identifies the certificate request in the keystore.
5. Type a common name (CN) value. This value is the CN value in the certificate distinguished name (DN).
6. You can configure one or more of the following optional values:
   a. Optional: Select a key size value. The valid key size values are 512, 1024, 2048, 4096, and 8192. The default key size value is 2048 bits.
   b. Optional: Type an organization value. This value is the O value in the certificate DN.
   c. Optional: Type an organizational unit value. This organizational unit value is the OU value in the certificate DN.
   d. Optional: Type a locality value. This locality value is the L value in the certificate DN.
   e. Optional: Type a state or providence value. This value is the ST value in the certificate DN.
   f. Optional: Type a zip code value. The zip code value is the POSTALCODE value in the certificate DN.
   g. Optional: Select a country value from the list. This country value is the C= value in the certificate request DN.
7. Click **Apply**.

Results

The certificate request is created in the specified file location in the keystore. The request functions as a temporary placeholder for the signed certificate until you manually receive the certificate in the keystore.

Note: Key store tools (such as iKeyman and keyTool) cannot receive signed certificates that are generated by certificate requests from WebSphere Application Server. Similarly, WebSphere Application Server cannot accept certificates that are generated by certificate requests from other keystore utilities.

What to do next

Now you can receive the CA-signed certificate into the keystore to complete the process of generating a signed certificate for your server.

Certificate request settings

Use this page to verify the properties of a personal certificate request.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store***. Under Additional Properties, click **Personal certificate requests > *certificate request*** .

File for certificate request:

Specifies the fully qualified name of the file that contains the certificate request that can be sent to a certificate authority (CA) server.

Key label:

Specifies the certificate alias name for the signer in the key store.

Key size:

Specifies the size of the keys that are generated.

Requested by:

Specifies the Subject distinguished name (DN) that represents the identity of the certificate request.

Fingerprint (SHA Digest):

Specifies the SHA hash of the personal certificate, which can be used to verify that the certificate has not been altered when it is used in a remote connection.

Signature algorithm:

Specifies the algorithm used to sign the certificate.

Personal certificates collection

Use this page to manage personal certificates.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store*** . Under Additional Properties, click **Personal certificates**.

The **Personal certificates** page lists all personal certificates in the selected key store. You can do most certificate management operations in this panel, including creating a new self-signed certificate, deleting a certificate, receiving one generated from a CA, replacing a certificate (simultaneous delete and create, replacing references across all key stores), extracting the signer, and importing or exporting a personal certificate.

Personal certificate requests are temporary place holders for certificates that will be signed by a certificate authority (CA).

The Key store collection must contain at least two key store files. You must select one file in order to replace, extract, or export a key store,

Table 159. Personal certificates buttons. This table lists the personal certificates buttons.

Button	Resulting action
Create (drop-down list)	Enables the application server to create the following certificates: • Self-signed Certificate • CA-signed Certificate • Chained Certificate
Delete	Specifies to delete a certificate from the key store. Be careful that the certificate alias is not referenced elsewhere in the Secure Sockets Layer configuration.
Receive a certificate from a certificate authority	Enables the application server to receive a certificate authority (CA)-generated certificate from a file to complete a certificate request.
Replace	Replaces a personal certificate with another personal certificate. All key stores in the configuration looking for signer certificate form the original personal certificate and replaces them with the new personal certificates signer. Any place in the security configuration where the certificate alias is referenced will be replaced with the new certificate alias.
Extract	Extracts the signer part of personal certificate from the key store and stores it to a file. The file can then be used to add the signer to another key store.
Import	Imports a certificate, including the private key, from a key store file or managed key store.

Table 159. Personal certificates buttons (continued). This table lists the personal certificates buttons.

Button	Resulting action
Export	Exports a certificate, including the private key, to a specified key store file or manage key store.
Revoke	Revokes a CA-signed certificate.
Renew	Renews a self signed or chained certificate.

Alias:

Specifies the alias by which the personal certificate is referenced in the key store.

When you select an alias, the View Certificate panel opens.

Issued by:

Specifies the distinguished name of the entity by which the certificate was issued. This name is the same as the issued-to distinguished name when the personal certificate is self-signed.

Issued to:

Specifies the distinguished name of the entity to which the certificate was issued.

Serial number:

Specifies the certificate serial number that is generated by the issuer of the certificate.

Expiration:

Specifies the expiration date of the signer certificate for validation purposes.

Self-signed certificates settings

Use this page to create self-signed certificates.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates >** *keystore*. Under Additional Properties, click **Personal certificates > Create (drop-down list) > Self-signed certificate**.

This same help file is available when you create a new certificate or view an existing certificate. The fields in this help file are described according to how they appear and are used on the administrative console.

Alias:

Specifies the alias for the personal certificate in the keystore.

You enter the alias name for the personal certificate in the keystore when you are creating a certificate. *The alias name is read-only when you view an existing certificate.*

Data type: Text

Version:

Specifies the version of the personal certificate. Valid versions include X509 V3, X509 V2, or X509 V1. It is recommended to use X509 V3 certificates.

This field is read-only when you create or view a certificate.

Data type: Text
Default: X509 V3
Range:

Key size:

Specifies the key size of the private key that is used by the personal certificate.

When you are creating a certificate you can select the key size from the drop-down list. *This field is read-only when you view a certificate.*

Data type: Integer
Default: 1024
Other valid key sizes: 512, 2048, 4096

Common name:

Specifies the common name portion of the distinguished name (DN). It is recommended that this name be the host name of the machine on which the certificate resides. In some cases, the common name is used to login during Secure Socket Layer (SSL) certificate authentication; therefore, in some cases, this name might be used as a user ID for a local operating system registry.

When you create a new certificate you can enter the common name in this field. *This field does not display when you view an existing certificate.*

Data type: Text

Serial number:

Identifies the certificate serial number that is generated by the issuer of the certificate. When creating a certificate this field does not appear.

This field is read-only when you view an existing certificate.

Validity period:

Specifies the length in days during which the certificate is valid. The default is 365 days. You can enter any number of days you wish.

This field is read-only when you view an existing certificate. This field displays a validity period as a range of days between two dates. For example, Valid from March 16, 2008 to March 16, 2009.

Data type: Text

Organization:

You enter the organization portion of the distinguished name. This field is optional.

This field displays only when you create a new certificate.

Data type: Text

Organization unit:

Specifies the organization unit portion of the distinguished name. This field is optional.

This field displays only when you create a new certificate.

Data type: Text

Locality:

Specifies the locality portion of the distinguished name. This field is optional.

This field displays only when you create a new certificate.

Data type: Text

State/Province:

Specifies the state portion of the distinguished name. This field is optional.

This field displays only when you create a new certificate.

Data type: Text

Zip code:

Specifies the zip code portion of the distinguished name. This field is optional.

This field displays only when you create a new certificate.

Data type: Integer

Country or region:

Select the country portion of the distinguished name from the drop-down list. This field is optional.

This field displays only when you create a new certificate.

Data type: Text
Default: (none)

Refer to http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html for a list of ISO 3166 country codes.

Validity period:

Identifies the length, in days, when the certificate is valid. The default is 365 days.

This field is read-only when you view an existing certificate and shows the start and end dates.

Issued to:

Identifies the distinguished name of the entity to which the certificate was issued.

This field is read-only when you view an existing certificate.

Issued by:

Identifies the distinguished name of the entity that issued the certificate. When the personal certificate is self-signed, this name is identical to the **Issued to** distinguished name.

This field is read-only when you view an existing certificate.

Fingerprint (SHA Digest):

Identifies the Secure Hash Algorithm (SHA hash) of the certificate, which can be used to verify the certificate's hash at another location, such as the client side of a connection.

This field is read-only when you view an existing certificate.

Signature algorithm:

Identifies the algorithm used to sign the certificate.

This field is read-only when you view an existing certificate.

Personal certificate requests collection

Use this page to manage personal certificate requests. Personal certificate requests are temporary place holders for certificates that will be signed by a certificate authority (CA).

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store*** . Under Additional Properties, click **Personal certificate requests**.

A private key is generated during the certificate request generation, but only the certificate is sent to the CA. The CA generates a new certificate, signed by the CA. This can be added in the Personal Certificates panel.

Table 160. Personal certificate requests buttons. This table lists the personal certificate requests buttons.

Button	Resulting action
New	Creates a personal certificate request that can be given to a certificate authority to complete.
Delete	Deletes a personal certificate request.
Extract	Extracts a personal certificate request. Only one certificate request can be selected at a time.
Query	Queries a personal certificate request. Only one certificate request can be selected at a time.

Note: Any changes made to this panel are permanent.

Key label:

Specifies the alias that represents the personal certificate request in the key store.

Requested by:

Specifies the Subject distinguished name (DN) that represents the identity of the certificate request.

Personal certificate requests settings

Use this page to create a new certificate request that can be extracted and sent to a certificate authority (CA).

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} >**

ssl_configuration. Under Related items, click **Key stores and certificates** > ***key store*** . Under Additional Properties, click **Personal certificates requests**. Then click the **New** button.

Personal certificate requests are temporary place holders for certificates that will be signed by a certificate authority (CA). The private key is generated during the certificate request generation, but only the certificate is sent to the CA. The CA generates a new certificate, signed by the CA.

Note: Any changes made to this panel are permanent.

File for certificate request:

Specifies the fully qualified file name from which the certificate request is exported. This portion of the certificate request can be given to the certificate authority to generate the real certificate. After the real certificate is generated, you can perform a "Receive a certificate from a certificate authority" from the personal certificate collection view.

Data type: String

Key label:

Specifies the alias that represents the personal certificate request in the key store.

Data type: String

Key size:

Specifies the size of the keys that are generated.

Data type: Integer
Default: 2048

Common name:

Specifies the name of the entity that the certificate represents. This common name can represent a person, company, or machine. For websites, the common name is frequently the DNS host name where the server resides.

Data type: String

Organization:

Specifies the organization portion of the distinguished name.

Data type: String

Organizational unit:

Specifies the organization unit portion of the distinguished name. This field is optional.

Data type: String

Locality:

Specifies the locality portion of the distinguished name. This field is optional.

Data type: String

State or province:

Specifies the state portion of the distinguished name. This field is optional.

Data type: String

Zip code:

Specifies the zip code portion of the distinguished name. This field is optional.

Data type: Integer

Country or region:

Specifies the country portion of the distinguished name.

Data type: String

Refer to http://www.iso.org/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html for a list of ISO 3166 country codes.

Extract certificate request

Use this page to extract a certificate request to a file so it can be sent to a certificate authority (CA).

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store*** . Under Additional Properties, click **Personal certificate requests > Extract**.

Key label:

Specifies the alias that represents the personal certificate request in the key store.

File for certificate request:

Specifies the filename where the extracted certificate request is placed.

Data type: Text

Receiving a certificate issued by a certificate authority

When a certificate authority (CA) receives a certificate request, it issues a new certificate that functions as a temporary placeholder for a CA-issued certificate. A keystore receives the certificate from the CA and generates a CA-signed personal certificate that WebSphere Application Server can use for Secure Sockets Layer (SSL) security.

Before you begin

The keystore must contain the certificate request that was created and sent to the CA. Also, the keystore must be able to access the certificate that is returned by the CA.

Note: To receive a certificate by using the wsadmin tool, use the **receiveCertificate** command of the AdminTask object. For more information, see the PersonalCertificateCommands command group for the AdminTask object article.

About this task

WebSphere Application Server can receive only those certificates that are generated by a WebSphere Application Server certificate request. It cannot receive certificates that are created with certificate requests from other keystore tools, such as **iKeyman** and **keyTool**.

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key stores and certificates >** [*keystore*].
2. Under Additional Properties, click **Personal certificates**.
3. Select a personal certificate.
4. Click **Receive a certificate from a certificate authority**.
5. Type the full path and name of the certificate file.
6. Select a data type from the list.
7. Click **Apply** and **Save**.

Results

The keystore contains a new personal certificate that is issued by a CA. The original certificate request is changed to a personal certificate.

What to do next

The SSL configuration is ready to use the new CA-signed personal certificate.

Export certificate to a keystore file or a managed keystore:

Use this page to specify a personal certificate to export to a keystore file or a managed keystore.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *keystore*** . Under Additional Properties, click **Personal certificates**. Select a personal certificate using the check box. Then click the **Export** button.

Certificate alias to export:

Displays the name of the certificate that you selected to export on the previous panel.

Data type: Text

Keystore Password:

Type in the password of the keystore to use for the export.

Data type: Text

Alias:

Specifies the alias that the personal certificate is referenced by in the destination keystore.

Data type: Text

Managed key store:

Select this option with the radio button. Then select a keystore from the pull-down list, which is managed by the security configuration, to export the certificate to.

Data type: Drop-down list

Key file name:

Select this option with the radio button. Then type the keystore file name into which the exported certificate is added. If the keystore file name already exists, the exported certificate is added. If the keystore file name does not already exist, a keystore file name is created, and the exported certificate is added.

Data type: Text

Type:

Specifies the type of keystore file. The valid types are listed in the drop-down list.

Data type: Text
Default: PKCS12

Key file password:

Specifies the password that is used to access the keystore file.

Data type: Text

Import certificate from a key file or managed keystore:

Use this page to specify a personal certificate to import from a keystore or key file.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *keystore*** . Under Additional Properties, click **Personal certificates**. Select a personal certificate using the check box. Then click the **Import** button.

Managed key store:

Select this option with the radio button. This selection indicates that the keystore that contains the certificate to import is a managed keystore.

Data type: radio button

Get key store aliases:

Clicking this button queries the configuration for the list of keystore aliases for which the certificate will be imported to.

Key store:

Select an alias of the keystore from the pull-down list of managed keystores that are managed by the security configuration. The alias you select identifies the keystore that contains the certificate to import.

Data type: drop-down list

Key store password:

Specifies the password for the keystore to use for import.

Data type: Text

Key store file:

Select this option with the radio button. This selection indicates a keystore file that contains the certificate to import.

Data type: radio button

Key file name:

Specifies the fully qualified path to keystore file that contains the certificate to import.

Data type: Text

Get key file aliases:

Clicking this button, queries the key file for the aliases of all the personal certificates in the keystore from which to choose.

Type:

Specify the type of keystore file. Select a valid type from the drop-down list.

Data type: Text

Key file password:

Type the password that is used to access the keystore file.

Data type: Text

Certificate alias to import:

Specifies the certificate alias identified as the **Key file name** that you want to import into the current keystore.

Data type: Text
Default: (none)

Imported certificate alias:

Specifies the new alias that you want the certificate to be named in the current keystore.

Data type: Text

Receive certificate from CA:

Use this page to import your personal certificate from the certificate authority (CA). The imported certificate replaces the temporary certificate associated with the public/private keys in the certificate request that is stored in the key store.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items click **Key stores and certificates** > ***key store*** . Under Additional Properties, click **Personal certificates > Receive certificate from certificate authority**.

Certificate file name:

Specifies the filename that contains the certificate generated by the certificate authority (CA).

Data type: Text

Data type:

Specifies the format of the file that is either Base64 encoded ASCII data or Binary DER data.

Data type: Text
Default: Base64-encoded ASCII data

Replace a certificate

Use this page to specify two certificates: the first selected certificate is replaced by the second selected certificate. The replace function replaces all the old signer certificates in key stores that are managed throughout the cell with the new signer from the new certificate. The same level of trust that was established with the old certificate is maintained. All places the certificate's alias is referenced in the security configuration will be replaced with the certificate's alias. The alias could be referenced on a security object like the SSL configuration, the dynamic outbound endpoint SSL configuration and key set groups.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates** > ***key store*** . Under Additional Properties click **Personal certificates**. Select a personal certificate, then click the **Replace** button.

Old certificate:

Specifies the certificate that you want to replace.

Data type: Text

Replace with:

Specifies the certificate that you want to replace the old certificate.

Data type: Text
Default: (none)

Delete old certificate after replacement:

Specifies that you want to delete the old certificate and all associated signer certificates after the new certificate replaces it. If you do not replace the old personal certificate, it might be assigned a new alias name.

Default: Disabled

Delete old signers:

Specifies that you want to delete the old signer certificates that are associated with the old certificate after the new signer certificates replace them. If you do not delete the old signer certificates, they might be assigned new alias names.

Default: Disabled

Extracting a signer certificate from a personal certificate

Personal certificates contain a private key and a public key. You can extract the public key, called the *signer certificate*, to a file, then import the certificate into another keystore. The client requires the signer portion of a personal certificate for Security Socket Layer (SSL) communication.

Before you begin

The keystore that contains a personal certificate must already exist.

Alternative Method: To extract a signer certificate from a personal certificate using the wsadmin tool, use the **extractCertificate** command of the AdminTask object. For more information, see the PersonalCertificateCommands command group for the AdminTask object article.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key stores and certificates > *keystore*** .
2. Under Additional Properties, click **Personal certificates**.
3. Select a personal certificate.
4. Click **Extract**.
5. Type the full path for the certificate file name. The signer certificate is written to this certificate file.
6. Select a data type from the list.
7. Click **Apply**.

Results

The signer portion of the personal certificate is stored in the file that is provided.

What to do next

This signer can now be imported into other keystores.

Extract certificate

Use this page to extract the signer from the personal certificate and store it in a file. The certificate can be added to a trust store for trust verification. When extracting the signer from a chained personal certificate, the signer at the top level of the chain is extracted.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store*** . Under Additional Properties, click **Personal certificates > Extract**.

Certificate alias to extract:

Displays the name of the certificate that you selected for extraction on the previous panel.

Data type: Text

Certificate file name:

Specifies the fully qualified path where the certificate file will reside.

Data type: Text

Data type:

Specifies the format of the file, which is either Base64-encoded ASCII data or Binary DER data.

Data type: Text
Default: Base64-encoded ASCII data

Extract signer certificate

Use this page to extract a signer certificate from the keystore to a file so that it can be added elsewhere.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store***. Under Additional Properties, click **Signer certificates** . Select a signer certificate, then click the **Extract** button

File name:

Specifies the fully qualified file name where the extracted signer certificate is placed.

Data type: Text

Data type:

Specifies the format of the file, which is either Base64 encoded ASCII data or Binary DER data.

Data type: Text

Retrieving signers using the retrieveSigners utility at the client

The client requires the signer certificates from the server to be able to communicate with WebSphere Application Server. Use the **retrieveSigners** command to get the signer certificate from a server.

Before you begin

The retrieveSigners utility is located in one of the following directories, depending on your operating system:

- *profile_root*/bin

In this release, a Java client that does not have access to a stdin console prompt should use the retrieveSigners utility to download the signers from the remote server key store when signers are needed for a Secure Sockets Layer (SSL) handshake. For example, you might interpret the client as failing to respond if an applet client or Java Web Start Client application cannot access the stdin signer exchange prompt. Thus, you must add the WebSphere Java method call **com.ibm.wsspi.ssl.RetrieveSignersHelper.callRetrieveSigners** to your client application to retrieve the signers and to avoid running the retrieveSigners utility manually.

Use the retrieveSigners utility for situations where you cannot verify whether or not the com.ibm.ssl.enableSignerExchangePrompt= property is enabled or disabled when the application makes a request. Set the com.ibm.ssl.enableSignerExchangePrompt= property to `false` in the ssl.client.props file if you cannot see the console.

Alternatively, you can manually create the server key in the client truststore.

About this task

Complete the following steps, as required:

Procedure

1. Use the **retrieveSigners** command to get the signer certificate from a server. You can find details about the **retrieveSigners** parameters in “Secure installation for client signer retrieval in SSL” on page 1727.
2. If the client and server are on the same machine, you will need only the *remoteKeyStoreName* and *localKeyStoreName* parameters. The most typical key store to reference on a remote system is `CellDefaultTrustStore` on a network deployed environment and `NodeDefaultTrustStore` on an application server.
3. When retrieving signers from a remote server, add these required connection-related parameters: **–host** *host*, **–port** *port*, **–conntype** {*RMI* | *SOAP*}.
4. Use the **–autoAcceptBootstrapSigner** parameter if you want to enable automation of the signer retrieval. This parameter automatically adds to the server all the signers that are needed to make the connection.

Results

After running, the command displays the SHI-1 digest of the signers added. The output looks similar to the following output:

```
/QIBM/UserData/WebSphere/AppServer/V67/Express/profiles/AppSrv01/bin/retrieveSigners
CellDefaultTrustStore ClientDefaultTrustStore

CWPKI0308I: Adding signer alias "default_signer" to local keystore
          "ClientDefaultTrustStore" with the following SHA digest:
```

Example

The following examples illustrate how to call the `retrieveSigners.bat` file.

To retrieve signers on the same system, enter:

```
profile_root/bin/retrieveSigners CellDefaultTrustStore ClientDefaultTrustStore
```

To retrieve signers on a remote system with a SOAP connection, enter:

```
profile_root/bin/retrieveSigners CellDefaultTrustStore ClientDefaultTrustStore
-host myRemoteHost -port 8879 -conntype SOAP -autoAcceptBootstrapSigner
```

To retrieve signers on a remote system that has security enabled, enter:

```
profile_root/bin/retrieveSigners CellDefaultTrustStore ClientDefaultTrustStore
-host myRemoteHost -port 8879 -conntype SOAP -user testuser -password testuserpwd
-autoAcceptBootstrapSigner
```

Changing the signer auto-exchange prompt at the client

For clients to communicate with WebSphere Application Server, clients must obtain a signer certificate from the server. Clients can use the **retrieveSigners** command to connect to a server to obtain the appropriate signer. A prompt displays that asks whether or not you want to add a signer to the truststore. If the Secure Sockets Layer (SSL) configuration uses an automated script that might hang, use the prompt to obtain the certificate.

Before you begin

The com.ibm.ssl.enableSignerExchangePrompt property in the *profile_home*/properties/ssl.client.props file controls the signer certificate prompt. By default, this property is set to `true`, meaning the prompt is enabled.

About this task

Complete the following steps to disable or enable the signer-exchange prompt at the client:

Procedure

1. Open the *profile_home*/properties/ssl.client.props file using an editor.
2. Locate the section containing the SSL configuration information for the client that you are working with.
3. Change the value of the com.ibm.ssl.enableSignerExchangePrompt property to `false` if you do not want the signer-exchange prompt, or set it to `true` if you want to be prompted.
4. Save and close the file.

Results

When the com.ibm.ssl.enableSignerExchangePrompt property is set to `false`, no prompt displays if a signer is not trusted. In this case the SSL handshake fails. Once the proper signer for the connection being made is manually installed in the trust store, the SSL handshake can succeed.

When the com.ibm.ssl.enableSignerExchangePrompt property is set to `gui` or `true`, a signer-exchange window is displayed, and you are asked to accept or reject the certificate. If you accept the certificate, it is installed in the trust store automatically and the handshake succeeds. If you reject the certificate, it does not get installed in the trust store and the handshake fails since the certificate is not trusted.

When the com.ibm.ssl.enableSignerExchangePrompt property is set to `stdin`, a signer-exchange ASCII prompt is displayed, and you are asked to accept or reject the certificate. If you accept the certificate, it is installed in the trust store automatically and the handshake succeeds. If you reject the certificate, it does not get installed in the trust store and the handshake fails since the certificate is not trusted.

The prompt looks like the following example:

Example

```
/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/default/bin/serverStatus -all
ADMU0116I: Tool information is being logged in file
/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/default/logs/serverStatus.log
ADMU0128I: Starting tool with the default profile
ADMU0503I: Retrieving server status for all servers
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1

*** SSL SIGNER EXCHANGE PROMPT ***
SSL signer from target host 192.174.1.5 is not found in truststore
/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/default/etc/trust.p12.
```

Verify that the digest value matches what is displayed at the server in the following signer information:

```
Subject DN:    CN=hostname.austin.ibm.com, O=IBM, C=US
Issuer DN:     CN=hostname.austin.ibm.com, O=IBM, C=US
Serial number: 1128544457
Expires:       Thu Oct 20 15:34:17 CDT 2006
SHA-1 Digest:  91:A1:A9:2D:F2:7D:70:0F:04:06:73:A3:B4:A4:9C:56:9D:A8:A3:BA
MD5 Digest:    88:72:C5:88:00:1C:A7:FA:D6:EB:04:88:AC:A1:C9:13

Add signer to the truststore now? (y/n) y
A retry of the request might need to occur.
ADMU0508I: The Application Server "server1" is STARTED.
```

What to do next

Clients can instigate communications for various processes using signer certificates obtained from WebSphere Application Server.

Retrieving signers from a remote SSL port

To perform Secure Sockets Layer (SSL) communication with a server, WebSphere Application Server must retrieve a signer certificate from a secure remote SSL port during the handshake. After the signer certificate is retrieved, you can add the signer certificate to a keystore.

Before you begin

The keystore that is to contain the signer certificate must already exist.

Alternative Method: To retrieve a signer certificate from a port using the wsadmin tool, use the **retrieveSignerFromPort** command of the AdminTask object. For more information, see the SignerCertificateCommands command group for the AdminTask object article.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > Key stores and certificates > *keystore* > Signer certificates > Retrieve from port**.
2. Click **Retrieve from port**.
3. Type the host name of the machine on which the signer resides.

4. Type the port location on the host machine on which the signer resides. The port location is not limited to ports on WebSphere Application Server. The ports can include Lightweight Directory Access Protocol (LDAP) ports or ports on any server on which an SSL port is already configured, such as `SIB_ENDPOINT_SECURE_ADDRESS`.
5. Select an SSL configuration for the outbound connection from the list.
6. Type an alias name for the certificate.
7. Click **Retrieve signer information**. A message window displays information about the retrieved signer certificate, such as: the serial number, issued-to and issued-by identities, SHA hash, and expiration date. If a chained certificate is on the port, information about the root is displayed.
8. Click **Apply**. This action indicates that you accept the credentials of the signer.

Results

The signer certificate that is retrieved from the remote port is stored in the keystore.

What to do next

An SSL configuration or client process that requires an SSL connection to the server can use the retrieved and approved signer certificate.

Retrieve from port

Use this page to retrieve a signer certificate from a remote SSL port. The system connects to the specified remote SSL host and port and receives the signer during the handshake using an SSL configuration.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store***. Under Additional Properties, click **Signer certificates**. Then click the **Retrieve from port** button.

To retrieve a signer certificate from a specific port, you enter the host and port, select an SSL configuration from the pull-down list, and enter an alias to identify the signer certificate. Click **Retrieve Signer Information** and information about the signer certificate is displayed, such as the serial number of the certificate, who the certificate is issued to and by, the certificate finger print, and the expiration information for the certificate. If you want the certificate to be stored in the keystore, click **Apply** or **Save**.

Host:

Specifies the host name to which you connect when attempting to retrieve the signer certificate from the Secure Sockets Layer (SSL) port.

Data type: Text

Port:

Specifies the SSL port to which you connect when attempting to retrieve the signer certificate.

Data type: Text

SSL configuration for outbound connection:

Specifies the SSL configuration that is used to connect to the previously specified SSL port. This configuration is also the SSL configuration that contains the signer after retrieval. This SSL configuration does not need to have the trusted certificate for the SSL port as it is retrieved during validation and presented here.

Data type: Text

Alias:

Specifies the certificate alias name that you want to reference the signer in the key store, which is specified in the SSL configuration.

Data type: Text

Retrieved signer information:

Specifies the signer certificate information if it is retrieved from the remote host and port.

Adding a signer certificate to a keystore

Signer certificates establish the trust relationship in SSL communication. You can extract the signer part of a personal certificate from a keystore, and then you can add the signer certificate to other keystores.

Before you begin

The keystore that you want to add the signer certificate to must already exist.

Alternative Method: To add a signer certificate to a keystore by using the wsadmin tool, use the **addSignerCertificate** command of the AdminTask object. For more information, see the SignerCertificateCommands command group for the AdminTask object article.

Note: If the security custom property com.ibm.websphere.security.OverwriteAndReplaceOnImport is set to `true` then import certificate imports a certificate and overwrites an existing certificate. It then perform the certificate replace operation on that certificate. Typically, an existing certificate cannot be overwritten by a certificate that is being imported. The task also replaces all signer certificates from the original certificate and replaces them with the signer certificate from the new certificate that is being imported

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > Inbound | Outbound > *ssl_configuration* > Key stores and certificates**.
2. Select a ***keystore*** from the list of keystores.
3. Click **Signer certificates**.
4. Click **Add**.
5. Enter an alias for the signer certificate in the **Alias** field
6. Enter the full path to the signer certificate file in the **File name** field.
7. Select a data type from the list in the **Data type** field.
8. Click **Apply**.

Results

When these steps are completed, the signer from the certificate file is stored in the keystore. You can see the signer in the keystore files list of signer certificates. Use the keystore to establish trust relationships for the SSL configurations.

Add signer certificate settings

Use this page to add a signer certificate in a certificate file to the keystore in the security configuration.

To view this administrative console page, click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key stores and certificates > *keystore* > Signer certificates > Add**.

Alias:

Specifies the alias that is used to identify the signer certificate in the keystore.

Data type: String

File name:

Specifies the path to the filename where the signer certificate is located.

Data type: String

Data type:

Specifies the format of the file, which is either Base64 encoded ASCII data or Binary DER data.

Data type: String

Signer certificates collection

Use this page to manage signer certificates in key stores. Signer certificates are used by Java Secure Socket Extensions (JSSE) to validate certificates sent by the remote side of the connection during a Secure Sockets Layer (SSL) handshake. If a signer does not exist in the trust store that can validate the certificate sent, the handshake fails and generates an "unknown certificate" error.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store***. Under Additional Properties, click **Signer certificates**.

Table 161. Signer certificates buttons. This table lists the signer certificates buttons.

Button	Resulting action
Add	Adds a new trusted (signer) certificate.
Delete	Deletes an existing signer certificate.
Extract	Extracts a signer certificate from a personal certificate to a file.
Retrieve from port	Makes a test connection to an SSL port and retrieves the signer from the server during the handshake. The information from the certificate will be displayed so you can decide whether to trust it based upon the MD5 and/or SHA hash.

Alias:

Specifies the alias for this signer certificate in the key store.

Issued to:

Specifies the distinguished name of the entity that requested the certificate.

Fingerprint (SHA digest):

Specifies the Secure Hash Algorithm (SHA hash) of the certificate. This can be used to verify the hash for the certificate at another location, such as the client side of a connection.

Expiration:

Specifies the expiration date of the signer certificate for validation purposes.

Signer certificate settings

Use this page to verify the general properties of the selected signer certificate.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates > *key store***. Under Additional Properties, click **Signer certificates**. Then click on a signer certificate.

Alias:

Specifies the alias for this signer certificate in the key store.

Version:

Specifies the version of the personal certificate. Valid versions include X509 V3, X509 V2, or X509 V1.

Key size:

Specifies the key size of the public key used by the signer certificate.

Serial number:

Specifies the certificate serial number that is generated by the issuer of the certificate.

Validity period:

Specifies the begin and end dates of the certificate.

Issued to:

Specifies the distinguished name of the entity that requested the certificate.

Issued by:

Specifies the distinguished name of the entity that issued the certificate. This name is the same as the issued-to distinguished name when the signer certificate is self-signed.

Fingerprint (SHA Digest):

Specifies the Secure Hash Algorithm (SHA) hash of the certificate, which can be used to verify the hash for the certificate at another location such as the client side of a connection.

Signature algorithm:

Specifies the algorithm that is used to sign the certificate.

Adding a signer certificate to the default signers keystore

Signer certificates are added to a keystore on the client side of an SSL communication to establish trust with the server. There is common practice for keystores to have trust established when they are created.

The **DmgrDefaultSignersStore** on a deployment manager and the **NodeDefaultSignersStore** on a stand alone application server are created to hold signer certificates used to establish trust by default in newly create keystores.

Before you begin

The default signers key store is created during profile creation and contains the signer certificate of the server default root certificate. Additional signer certificates can be added to the default signers key store at any time. Anytime a keystore is created using the admin console or by using the **createKeyStore** AdminTask object in scripting, all signer certificates from the default signer store are added to the newly created keystore.

Alternative Method:

- To add a signer certificate to a default signer keystore by using the wsadmin tool, use the **addSignerCertificate** command of the AdminTask object.
- To create a new keystore by using the wsadmin tool, use the **createKeyStore** command of the AdminTask object.
- To extract the signer from a personal certificate using the wsadmin tool, use the **extractCertificate** of the AdminTask object.
- To exchange a signer certificate using the wsadmin tool, use the **KeyStoreCommands** command group for the AdminTask object.

For more information, see the SignerCertificateCommands command group for the AdminTask object article and the KeyStoreCommands command group for the AdminTask object article.

Procedure

1. If the certificate is in a certificate file, it can be added to the default signer keystore using the administrative console.
   a. Click **Security > SSL certificate and key management**.
   b. Under Related Items, click **Key stores and certificates**.
   c. c. Select Default signers keystore under KeyStore Usages. A panel displaying a list of keystores appears.
   d. Click on **DmgrDefaultSignersStore**.
   e. Under Additional Properties, click **Signer certificates**.
   f. Click **Add** .
   g. Enter an alias in the alias box, a path to the certificate file in the filename box, and an asterisk (•). Select the format of the certificate file from the pull down list in the “Data type” box.
   h. Click **Apply** then **Save**.

   Note: You can also perform this addition using the AdminTask, **addSignerCertificate**.
2. If the signer certificate form of a personal certificate needs to be added to default signers keystore, you can extract the signer from the personal certificate to a certificate file or the signer can be extracted directly to the default signers keystore. To extract a signer certificate from a personal certificate to a certificate file,
   a. Click **Security > SSL certificate and key management**.
   b. Under Related Items, click **Key stores and certificates**.
   c. c. Select All under Keystore Usages. A panel displaying a list of keystores appears.
   d. Click on the keystore name
   e. Under Additional Properties, click **Personal certificates**.
   f. Select a personal certificate.

g. Click **Extract**.
h. Enter the path to the certificate file in “Certificate file name” box and select a format type from the pull down list in “Data type” box
i. Click **Apply** then **Save**.
j. The signer can be added to the default signers keystore by following step 1.

Note: You can also extract the signer from a personal certificate using scripting and the AdminTask **extractCertificate**.

3. To extract a signer certificate to the default signers keystore, an exchange of the signer certificate can be performed from the administrative console.
   a. Click **Security > SSL certificate and key management**
   b. Under Related Items, click **Key stores and certificates**.
   c. c. Select All under Keystore Usages. A panel displaying a list of keystores appears.
   d. Click on the default signers keystore and the keystore that contains the personal certificate whose signer certificate is needed.
   e. Click **Exchange Signers**.
   f. Select the personal certificate whose signer is needed.
   g. Click **Add**.
   h. Click **Apply** then **Save**.

   Note: You can also perform the exchange using the AdminTask, **exchangeSigner**.

   Note: A DataPower certificate can be removed from the default signers keystore if it is present. If you are not using the DataPower appliance manager you should remove the DataPower certificate from the default trust store to avoid unintentional trust relationships. However, if you start to use DataPower appliance manager at a later date you must add the DataPower certificate back to the default trust store.

Results

When these steps are completed, the signer from the certificate file is stored in the default signers keystore. You can see the signer in the keystore files list of signer certificates.

What to do next

The new keystore will contain the default signers that were added to the default signers keystore.

Exchanging signer certificates

To establish trust relationships, you can exchange signer certificates between keystores. When you exchange signer certificates, you are extracting a personal certificate from one keystore and adding it to another keystore as a *signer certificate*.

Before you begin

To exchange signer certificates, there must be two keystores.

About this task

Complete the following steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key stores and certificates**.
2. Select two keystores from the list of keystores.
3. Click **Exchange signers**.
4. Select any of the certificates in the first personal certificates list, and click **Add**. After adding, the signer part of the selected personal certificate appears in the other (second) keystore signers list.
5. Select any of the certificates in the second personal certificates list, and click **Add**. After adding, the signer part of the selected personal certificate appears in the other (first) keystore signers list.
6. Optional: If you need to remove any of the certificates from either of the signers lists, highlight one or more of the certificates, and click **Remove**.
7. Click **Apply** and **Save**.

Results

The signer certificate appears in the list for each keystore.

What to do next

The extracted signer certificate is available to both keystores during the connection handshake.

Keystores and certificates exchange signers

Use this page to extract the signer part of a personal certificate from one keystore and add it to another keystore as a signer certificate. Signer certificates can also be listed, and they will be added to the other keystore as well.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key stores and certificates** then select two key stores to exchange and click the **Exchange signers**.

Note: Any changes made to this panel are permanent.

[keystore] personal certificates:

Specifies the personal certificates and signer certificates that are currently stored in the specified keystore.

Press and hold the **Ctrl** key to select more than one item from the list.

Data type: Text

[keystore] signers:

Specifies the trusted signer certificates that are currently stored in the specified keystore and selected for the exchange.

Press and hold the **Ctrl** key to select more than one item from the list.

Data type: Text

Add:

Specifies to extract the signer from the selected personal certificate in the keystore list on the left and add it to the signers list of the keystore on the right.

After the certificate is added, it no longer displays in the left-hand list. The personal certificate is still in the keystore, but it is no longer selectable

Remove:

Specifies to remove a selected signer from the signers list of the keystore on the right. The removed certificate displays in the keystore list on the left.

Configuring certificate expiration monitoring

When certificates expire, they can no longer be used by the system. WebSphere Application Server provides a utility to monitor certificates that are close to expiration or have already expired. You can schedule certificate monitoring, or you can request certificate monitoring on demand. You can also configure options for deleting expired certificates and for recreating certificates.

Before you begin

Important: The Certificate Expiration Monitor does not handle replacing client self-signed certificates and is not capable of sending the new signer certificate needed for trust. If the client is a web server plug-in, it will not be able to securely communicate with the application server after self-signed certificate replacement.

WebSphere Application Server notifies you when a certificate is about to expire. Complete the information required for notification messaging in “Notifications” on page 1833.

About this task

Complete the following configuration steps in the administrative console:

Procedure

1. Click **Security > SSL certificate and key management > Manage certificate expiration**.
2. Type a number for the number of days threshold in the **Expiration notification threshold** field. WebSphere Application Server issues an expiration warning *n* number of days before expiration.
3. Select or check one or more of the following options:
   - **Expiration check notification**. Select the method from the list that you want to use to receive your notification.
   - **Automatically replace expiring self-signed certificates**. If you do not want to recreate the self-signed certificate, clear the check box.

     Attention: When using writable System Authorization Facility (SAF) keyrings in your configuration, the certificate expiration monitor does not replace expired certificates in the writable SAF keyrings, but only provides a notification of the expiration.
   - **Delete expiring certificates and signers after replacement**. If you do not want to delete the expired certificates and signers, clear the check box.
   - **Enable checking**. If you do not want to have certificate monitoring enabled, clear the check box.
4. Enter the time of day when you want certificate monitoring to take place to schedule the running of the certificate expiration monitor.
5. Select one of the following options:
   - **Check by calendar**. For **Weekday**, enter the day of week that you want to run the certificate expiration monitor. For **Repeat Interval**, specify the frequency to run the certificate monitor.
   - **Check by number of days**. Enter a number for how frequently the monitor runs, in number of days.

6. Type the number of days before the threshold date in which the certificate monitor warns that a certificate is about to be replaced. When a certificate is within the expiration threshold, and automatic replacement is enabled, certificates are replaced. This value specifies the time period before the threshold when warnings are issued by the certificate monitor concerning upcoming replacement dates.
7. Click **Apply**.

Results

After completing the settings, a certificate expiration monitor object and a schedule are set up in the configuration. The certificate expiration monitor runs according to the configurations options that you configured.

What to do next

You can generate reports that state which certificates have expired. The reports identify the notifications of certificate replacements and deletions. The report is sent according to the notification option that you specified.

Manage certificate expiration settings

Use this page to configure the certificate expiration monitor.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage certificate expiration**.

Attention: To see the changes to the Expiration checking fields, you must click **Apply**.

Start now:

Specifies to start certificate monitoring. When the monitor runs, it visits all the key stores and checks to see if they are within certificate expiration range. If you set the option to delete or replace expired certificates, you can run these operations immediately by pressing **Start now**.

Expiration notification threshold:

Specifies the period of time that occurs chronologically just before the expiration day of the certificate, within which, if the ExpirationMonitor thread runs, and **Automatically replace expiring self-signed and chained certificates** is enabled, a new self-signed or chained certificate is generated. By default, the replacement period for the certificate is 60 days in length or less as defined in the daysBeforeNotification property.

There is a pre-notification period where the certificate is added to the notification list but not touched for 90 days prior to the 60 days. By default, this pre-notification period is 90 days in length as defined in the com.ibm.ws.security.expirationMonitorNotificationPeriod property.

Data type:	Integer
Default:	60 days or less

Enable checking:

Specifies the certificate monitor is active and will run as scheduled.

Scheduled time of day to check for expired certificates:

Specifies the scheduled time that the system checks for expired certificates.

You can type the scheduled time in hours and minutes, specify either A.M. or P.M., or 24-hour.

Data type Integer
Default: 0, 0
Range: 1–12, 0–59

Check by calendar:

Indicates that you want to schedule a specific day of the week on which the expiration monitor runs. For example, it might run on Sunday.

Default: Enabled

Weekday:

Specifies the day of the week on which the expiration monitor runs if **Check on a specific day** is selected.

Default: Sunday
Range: Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday

Repeat interval:

Specifies the period of time between each schedule time to check for expired certificates or the interval between schedule checks.

Default: Daily
Range: Daily, Weekly

Check by number of days:

Specifies that you want to schedule a specific number of days between each run of the expiration monitor. The day of the week on which this occurs is not counted. For example, if you set the interval to check for expired certificates every seven days, the expiration monitor runs on day eight.

Default: Disabled

Next start date:

Specifies the date for the next scheduled check. This allows the deployment manager to be stopped and restarted without resetting the date.

Expiration check notification:

Specifies the notification type (either email, or an entry in the system log) when an expiration monitor runs.

Default:

Automatically replace expiring self-signed certificates and chained certificates:

Specifies a new self-signed certificate or chained certificate be generated using the same certificate information if the expiration notification threshold is reached. The old certificate is replaced and uses the same alias. All old signers are managed by the key store configuration are also replaced. The system only replaces self-signed certificates.

Note: This checkbox is only applicable when using file based keystores.

Default: Enabled

Delete expiring certificates and signers after replacement:

Specifies whether to completely remove old, self-signed certificates from the key store during a replace operation or leave them there under a renamed alias. If an old certificate is not deleted, the system renames the alias so that the new certificate can use the old alias, which might be referenced elsewhere in the configuration.

Note: This checkbox is only applicable when using file based keystores.

Default: Enabled

Notifications

Use this page to specify the generic notification definitions that are used in certificate expiration monitors.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click**Manage certificate expiration**. Under Related items, click **Notifications**.

Table 162. Notifications buttons. This table lists the notifications buttons.

Button	Resulting action
New	Adds a notification. The notification configures how the expiration monitor notifies the administrator of certificates that will expire within the specified threshold.
Delete	Deletes an existing notification.

Notification name:

Specifies the notification name.

Message log:

Specifies that this configuration intends to log certificate expiration information to the message log file.

Send Email:

Specifies that this configuration intends to send certificate expiration information to the list of users in the email list.

List of email addresses:

Specifies the email addresses that are sent notifications when certificates fall within the expiration threshold. You must specify the SMTP server for each email address. If an email address is not specified, by default the application server assumes that the SMTP server is "smtp-server." For example, if you type *name*@domain, the SMTP server will be smtp-server.domain.

Notifications settings

Use this page to set properties for new notifications used in certificate expiration monitors or for security audit subsystem failures.

To view this administrative console page perform one of the following:

- Click **Security > SSL certificate and key management** > **Manage certificate expiration** >**Notifications > New**.
- Click **Security** > **Security auditing** > **Audit monitor** > **New**

.

Notification name:

Specifies the name of the notification configuration.

Data type: Text

Message log:

Specifies that this configuration will log the notification to a message log file.

Default: Disabled

Email sent to notification list:

Specifies that this configuration send a notification as an email to the email list.

Default: Disabled

Email address to add:

Specifies the email addresses that are sent notifications. You must specify the SMTP server for each email address. If an email address is not specified, by default the application server assumes that the SMTP server is "smtp-server." For example, if you type *name*@domain, the SMTP server will be smtp-server.domain.

Data type: Text (format as valid Internet mail address)

Add:

Adds the email address to the right-hand list.

Remove:

Removes the email address from the right-hand list.

Outgoing mail (SMTP) server:

Specifies the SMTP server to be used with the email address. If none is specified, the email realm will be used.

Key management for cryptographic uses

WebSphere Application Server provides a framework for managing keys (secret keys or key pairs) that applications use to perform cryptographic operations on data. The key management framework provides an application programming interface (API) for retrieving these keys. Keys are managed in keystores so the keystore type can be supported by WebSphere Application Server, provided that the keystores can store the referenced key type. You can configure keys and scope keystores so that they are visible only to particular processes, nodes, clusters, and so on.

The key management infrastructure is based on two key configuration object types: key sets and key set groups. WebSphere Application Server uses a key set to manage instances of keys of the same type. You can configure a key set to generate a single key or a key pair, depending on the key or key pair generator class. A key set group manages one or more key sets and enables you to configure and generate different

key types at the same time. For example, if your application needs both a secret key and key pair for cryptographic operations, you can configure two key sets, one for the key pair and one for the secret key that the key set group manages. The key set group controls the auto-generation characteristics of the keys, including the schedule. The framework can automatically generate keys on a scheduled basis, such as on a particular day of the week and time of day, so that key generation is done during off-peak hours.

Figure 1 shows an example of a key set group that is configured to manage two key sets: key set 1 and key set 2.

Figure 35.

Key set 1 generates key pairs. Key set 2 generates secret keys. The application needs both types of keys for its cryptographic operations, signing and encryption, on data. The keys for each key set need to be generated in tandem. The application stores the key set group name with the encrypted data. The key set group generates a new set of keys every Sunday night at 11 P.M.. The application maintains key generation data for two weeks.

Creating a key set configuration

You can use key sets to manage multiple instances of cryptographic keys. WebSphere Application Server uses keys to encrypt or sign outbound data, and decrypt or verify inbound data during cryptographic operations.

Before you begin

You must have write-access to the keystore that will contain the keys after you generate them from a key set. However, if you want to generate keys outside of WebSphere Application Server, you can reference the keys from a read-only keystore that contains a secret key that you can access when you generate the keys. If you are creating a key pair using an X509Certificate and a PrivateKey object , see “Example: Developing a key or key pair generation class for automated key generation” on page 1842.

About this task

Complete the following steps in the administrative console:

Procedure

1. Decide whether you want to create the key set at the cell scope or below the cell scope at the node, server, or cluster, for example:
   - To create a key set at the cell scope, click **Security > SSL certificate and key management > Key sets**.
   - To create a key set at a scope below the cell level, click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration* > Key sets**.
2. Click **New** to create a new key set.
3. Type a key set name. For example, `CellmyKey`.
4. Type a key alias prefix name. For example, `myKey`. This field specifies the prefix for the key alias when the new key is generated and stored in the keystore. Following the prefix is the key reference version number, for example, 2, so that the full key alias name would be `myKey_2`. If the key reference already has a specified alias for a key that exists in the keystore, then WebSphere Application Server ignores this field.
5. Type a key password. The key password protects the key in the keystore. This password is ignored by WebSphere Application Server if you already specified a password for the key alias reference. To

check for a key reference password, click **Active key history** under Additional Properties. The key reference password protects keys that are generated by a key generator class.

6. Type the password again to confirm it.
7. Optional: Type the key generator class name. For example, `com.ibm.ws.security.ltpa.LTPAKeyGenerator`. The class name generates keys. If the class implements com.ibm.websphere.crypto.KeyGenerator, then a getKey method returns a java.security.Key object that is set in the keystore using the setKey method without a certificate chain. If the class implements com.ibm.websphere.crypto.KeyPairGenerator, then a getKeyPair method returns a com.ibm.websphere.crypto.KeyPair object that contains either a java.security.PublicKey and java.security.PrivateKey or a java.security.cert.Certificate and a java.security.PrivateKey object. The key generator class and the KeySetHelper API specify the details of the keys that are generated.
8. Optional: Select **Delete key references that are beyond the maximum number of keys** if you do not want old keys saved in the keystore after WebSphere Application Server removes their references from the Active key history listing. The Active key history lists the keys that the KeySetHelper API is currently tracking. The number of keys in the list is equal to the number of keys that you specify in **Maximum number of keys referenced**.
9. Type a numeric value for the maximum number of keys referenced. For example, if you type `3` and select **Delete key references that are beyond the maximum number of keys**, the fourth key version generation automatically triggers WebSphere Application Server to delete the first key version from the keystore. If you choose not to delete the old keys, they do not display in the Active key history list but instead remain in the keystore where you can remove them manually.
10. Select a keystore from the drop-down list.
    - Select a JCEKS keystore if you are storing a secret key.
    - Select any keystore if you are storing a key pair with an X509Certificate and PrivateKey object.
11. Optional: Select **Generates key pair** if your key generator class name implements the com.ibm.websphere.crypto.KeyPairGenerator interface instead of the com.ibm.websphere.crypto.KeyGenerator interface. This option designates that the key references a key pair instead of a single key. A key pair contains both a public key and a private key. The WebSphere Application Server run time determines whether or not key pairs are stored and loaded differently than single keys.
12. Optional: Click **Apply** if you want to select **Active key history** under Additional Properties to add alias references or generate more keys.
    a. Click **Active key history**.
    b. Click **Add key alias reference** if you are not using the key generator class name to add key alias references to the keys that already exist in the keystore. Use this option to retrieve the keys from a read-only keystore without the key set generating them.
    c. Type an alias reference.
    d. Click **Generate key** if you want to generate a key using the class name that you defined in the key sets panel. Each new key increments numerically, for example, `myAlias_2`.
    e. Click **Apply**.
13. Click the key set name in the navigation path at the top of the panel.
14. Click **OK** and **Save**.

Results

You have created a key set that you can manage using the **Active key history** link. You can generate keys manually to associate them with specified key sets.

What to do next

After you generate new keys from a key set, you can access them programmatically using the com.ibm.websphere.crypto.KeySetHelper API. You must have Java 2 Security permissions, if enabled, to

access keys in key sets. Specify the key set name within the fine-grained permissions, as in the following code sample: `WebSphereRuntimePermission "getKeySets.keySetName"`. For more information, see "Example: Retrieving the generated keys from a key set group" on page 1841. To generate multiple key types at the same time or to schedule the key generation on a specific schedule, see "Creating a key set group configuration" on page 1840.

Active key history collection

Use this page to manage key alias references.

To view this administrative console page, click **Security** > **SSL certificate and key management** > **Manage endpoint security configurations > {Inbound | Outbound}** > ***ssl_configuration*** > **Key Sets** > ***key set*** > **Active key history**.

Table 163. Active key history buttons. This table lists the active key history buttons.

Button	Resulting action
Add key alias reference	Adds a reference to a key that already exists in a key store. If a key generation class is configured, the references are added automatically during generation and do not need to be added manually.
Delete	Deletes an existing key reference. This action does not delete the key in the keystore.
Generate key	Generates a key. The button is displayed only if a generator class name is specified for the key set, and the selected key store is editable.

Alias reference:

Specifies the name of the alias as it appears in the keystore.

Add key alias reference settings

Use this page to access key alias reference information.

To view this administrative console page, Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key Sets > *key set***. Under Additional Properties, click **Active key history** then click the **Add key alias reference** button.

Alias reference:

Specifies the name of the alias as it appears in the key store.

Data type: Text

Password:

Specifies the key password to get access to the key. This password is enforced by the keystore for that specific key. If the key does not have a password, this field can be left blank.

Data type: Text

Confirm password:

Confirms the password entered in the previous field.

Data type: Text

Key sets collection

Use this page to manage key sets, which control a set of key instances of the same type for use in cryptographic operations. The keys can either be generated using a custom class or reference keys that already exist in a keystore.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key sets**.

Table 164. Key set buttons. This table lists the key set buttons.

Button	Resulting action
New	Adds a new key set.
Delete	Deletes an existing key set. Make sure the key set is not referenced by a key set group before deleting it.

Key set name:

Specifies the key set name that is used to select the key set from a key set group and from runtime application programming interfaces (API).

Key store:

Specifies the key store that contains the keys for storage, retrieval, or both.

Key alias prefix name:

Specifies the prefix for the key alias when a new key is generated and stored in a key store. The rest of the key alias comes from the key reference version number.

For example, if the alias prefix is `mykey` and the key reference version is 2, the keystore references the key using alias `mykey_2`. If the key reference already has a specified alias for a key already existing in the keystore, this field is ignored.

Key sets settings

Use this page to set the properties for a new key set.

To view this administrative console page, click **Security** > **SSL certificate and key management**. Under Configuration settings, click**Manage endpoint security configurations** > **{Inbound | Outbound}** > ***ssl_configuration***. Under Related items, click **Key sets** > **New**.

Key set name:

Specifies the key set name that is used to select the key set from a key set group and from runtime application programming interfaces (API).

Data type: Text

Management scope:

Specifies the scope where this Secure Sockets Layer (SSL) configuration is visible. For example, if you choose a specific node, then the configuration is only visible on that node and any servers that are part of that node.

Data type List
Range: Applicable scopes

Key alias prefix name:

Specifies the prefix for the key alias when a new key is generated and stored in a keystore. The rest of the key alias comes from the key reference version number. For example, if the alias prefix is `mykey` and the key reference version is 2, the keystore references the key using alias `mykey_2`. If the key reference already has a specified alias for a key already existing in the keystore, this field is ignored.

Data type: Text

Key password:

Specifies the password used to protect the key in the keystore. If a password is specified in the key reference as well, this password is ignored. This password is used for keys that get generated by a key generator class.

Data type: Text

Confirm password:

Specifies the same password again to confirm it was entered correctly the first time.

Data type: Text

Key generator class name:

Specifies the class name that generates keys. If the class implements com.ibm.websphere.crypto.KeyGenerator, then a getKey() method should return a java.security.Key object that is set in the key store using the setKey method without a certificate chain. The key store type associated with the key set must support storing keys without certificates, such as JCEKS.

Data type: Text

If the class implements com.ibm.websphere.crypto.KeyPairGenerator, then a getKeyPair() method should return a com.ibm.websphere.crypto.KeyPair object containing either a java.security.PublicKey and java.security.PrivateKey, or a java.security.cert.Certificate[] and a java.security.PrivateKey. The key generator class and the caller of the KeySetHelper API should know the details of the keys that are generated. This framework does not need to understand the key algorithms and lengths.

Delete key references that are beyond the maximum number of keys:

Specifies that the keys are deleted from the keystore at the same time that the key reference is deleted. The server deletes the older key references as the Maximum number of keys referenced value is exceeded.

Maximum number of keys referenced:

Specifies the maximum number of key instances that are returned when keys from this key set are requested. The oldest key reference gets removed whenever a new key reference gets generated after the maximum has been reached.

Data type: Integer
Default: 3

Key store:

Specifies the key store that contains the keys for storage, retrieval, or both.

Data type: Text

Generates key pair:

Specifies that a key references a key pair instead of a key. The key pair contains both a public key and a private key.

Creating a key set group configuration

A key set group manages one or more key sets. WebSphere Application Server uses key set groups to automatically generate cryptographic keys or multiple synchronized key sets.

About this task

Complete the following steps in the administrative console:

Procedure

1. Decide whether you want to create the key set group at the cell scope or below the cell scope at the node, server, or cluster, for example.
   - To create a key set group at the cell scope, click **Security > SSL certificate and key management > Key set groups**.
   - To create a key set group at a scope below the cell level, click **Security > SSL certificate and key management > Manage endpoint security configurations > {Inbound | Outbound} > *SSL_configuration* > Key set groups**.
2. You can choose to generate a key for an existing key set group, delete an existing key set group, or create a new key set group.
   - To generate a key for an existing key set group, select a key set group from the list of existing key set groups, and click **Generate keys**. You have generated a new key for each key set in the selected group.
   - To delete an existing key set group, select a key set group from the list of existing key set groups, and click **Delete**. You have deleted the key set group.
   - To create a new key set group, go to step 3.

   CAUTION:
   Do not delete the cell or node LTPAKeySetGroup, which is used by the Lightweight Third Party Authentication (LPTA) mechanism.
3. Click **New** to create a new key set group.
4. Type a key set group name. You can reference this name by using the com.ibm.websphere.crypto.KeySetHelper API to retrieve the managed keys from an application.
5. Select one or more key sets from the Key sets list.

   Note: If the key set(s) you want is not listed, make sure that it was created at the same scope or a higher scope than where you are creating the new key set group.
6. Click **Add** to add the selected key set(s) to the new key set group.
7. Select **Automatically generate keys** to generate the new keys on a schedule. If you decide to generate keys automatically, then you must specify a scheduled time of day.
8. Specify the scheduled time to generate keys automatically in hours and minutes, A.M. or P.M., or every 24 hours.
9. You can choose to generate new keys on a specific day or at an interval.

- Select **Generate on a specific day**. Select a day of the week from the drop-down list, and type a repeat interval number for the number of days between each key generation. This choice enables you to schedule key generation when your systems are least busy.
- Select **Generate at an interval**. Type a repeat interval number for the number of days between each key generation. This choice enables you to schedule key generation more frequently than once a week.

Note: The **Next start date** is a read-only field that specifies the date for the next scheduled generation. You can stop and restart the deployment manager or base application server without resetting this date. If you do not see the next start date appear after changing the configuration, click **OK** to save it, then check that the next start date displays.

10. Click **Save**.

Results

You have created a new key set group to manage key sets and key generation on a schedule.

What to do next

After you generate new keys from a key set, you can access them programmatically using the com.ibm.websphere.crypto.KeySetHelper API. You must have Java 2 Security permissions, if enabled, to access keys in key sets. Specify the key set name within the fine-grained permissions, as in the following code sample: `WebSphereRuntimePermission "getKeySets.`*`keySetName`*`"`. For more information, see "Example: Retrieving the generated keys from a key set group."

Example: Retrieving the generated keys from a key set group

This example shows how applications can use the com.ibm.websphere.crypto.KeySetHelper API to retrieve managed keys from the KeySet or KeySetGroup configurations. Use the com.ibm.websphere.crypto.KeySetHelper API to get either the latest set of keys or all the keys in the KeySet or KeySetGroup object.

Use the latest keys when performing any new cryptographic operations. All of the other keys that are defined in the KeySet or KeySetGroup object are for the validation of previously performed cryptographic operations.

The following example uses a method that an application might use to initialize the keys in the associated KeySetGroup object. The application might want to store the keys in two separate maps, one for generation and one for validation. Refer to the API documentation for KeySetHelper API to determine which Java 2 Security requirements are required.

```
/**
     * Initializes the primary and secondary Maps used for initializing the keys.
     */

    public void initializeKeySetGroupKeys() throws com.ibm.websphere.crypto.KeyException
    {
        java.util.Map generationKeys = null;
        java.util.Map validationKeys = null;

        PublicKey tempPublicKey = null;
        PrivateKey tempPrivateKey = null;
        byte[] tempSharedKey = null;

        keySetGroupName = "ApplicationKeySetGroup";
        com.ibm.websphere.crypto.KeySetHelper ksh = com.ibm.websphere.crypto.KeySetHelper.getInstance();
        generationKeys = ksh.getLatestKeysForKeySetGroup(keySetGroupName);

        /***
        *  Latest keys: {
        *  KeyPair_3=com.ibm.websphere.crypto.KeyPair@64ec64ec,
        *  Secret_3=javax.crypto.spec.SecretKeySpec@fffe8aa7
        *  }
        ***/

        if (generationKeys != null)
        {
```

```
            Iterator iKeySet = generationKeys.keySet().iterator();

            while (iKeySet.hasNext())
            {
                String keyAlias = (String)iKeySet.next();

                Object key = generationKeys.get(keyAlias);

                if (key instanceof java.security.Key)
                {
                    tempSharedKey = ((java.security.Key)key).getEncoded();
                }
                else if (key instanceof com.ibm.websphere.crypto.KeyPair)
                {
                    java.security.Key publicKeyAsSecret =
    ((com.ibm.websphere.crypto.KeyPair)key).getPublicKey();
                    tempPublicKey = new PublicKey(publicKeyAsSecret.getEncoded());
                    java.security.Key privateKeyAsSecret =
    ((com.ibm.websphere.crypto.KeyPair)key).getPrivateKey();
                    tempPrivateKey = new PrivateKey(privateKeyAsSecret.getEncoded());
                }
            }

            // save these for use later, if necessary
            validationKeys = ksh.getAllKeysForKeySetGroup(keySetGroupName);

            /***
            *  All keys: {
            *  version_1=
            *      {Secret_1=javax.crypto.spec.SecretKeySpec@178cf,
            *       KeyPair_1=com.ibm.websphere.crypto.KeyPair@1c121c12},
            *  version_2=
            *      {Secret_2=javax.crypto.spec.SecretKeySpec@17a77,
            *       KeyPair_2=com.ibm.websphere.crypto.KeyPair@182e182e},
            *  version_3=
            *      {Secret_3=javax.crypto.spec.SecretKeySpec@fffe8aa7,
            *       KeyPair_3=com.ibm.websphere.crypto.KeyPair@4da04da0}
            *  }
            ***/
        }
        else
        {
            throw new com.ibm.websphere.crypto.KeyException("Could not generateKeys.");
        }
    }
```

Example: Developing a key or key pair generation class for automated key generation

A class that generates keys for cryptographic operations can be created automatically. With this capability, the key management infrastructure can maintain a list of keys for a predefined key set, and applications can access these keys.

You can schedule new key generation at predefined frequencies. Remember that key generation frequency affects the security of your data. For example, for persistent data, you might schedule key generation less frequently than for real time communications, which require that the keys be generated more often as old keys expire.

When you develop a key generation class, decide if you are creating a shared key or a key pair because this decision determines the interface you must use.

If you are developing shared keys, refer to the following example, which uses the KeyGenerator class to implement the com.ibm.websphere.crypto.KeyGenerator interface. The interface returns a java.security.Key key, which is stored as a SecretKey in a JCEKS keystore type. You can use any other keystore type that supports storing secret keys.

```
package com.ibm.test;

import java.util.*;
import com.ibm.ws.ssl.core.*;
import com.ibm.ws.ssl.config.*;
import com.ibm.websphere.crypto.KeyException;

public class KeyGenerator implements com.ibm.websphere.crypto.KeyGenerator
{
    private java.util.Properties customProperties = null;
    private java.security.Key secretKey = null;

    public KeyGenerator()
    {
```

```
    }

     /**
      * This method is called to pass any custom properties configured with
      * the KeySet to the implementation of this interface.
      *
      * @param java.util.Properties
      **/
    public void init (java.util.Properties customProps)
    {
        customProperties = customProps;
    }

     /**
      * This method is called whenever a key needs to be generated either
      * from the schedule or manually requested. The key is stored in the
      * KeyStore referenced by the configured KeySet that contains the
      * keyGenerationClass implementing this interface. The implementation of
      * this interface manages the key type. The user of the KeySet
      * must know the type that is returned by this keyGenerationClass.
      *
      * @return java.security.Key
      * @throws com.ibm.websphere.crypto.KeyException
      **/
    public java.security.Key generateKey () throws KeyException
    {
        try
        {
            // Assume generate3DESKey is there to create the key.
            byte[] tripleDESKey = generate3DESKey();
            secretKey = new javax.crypto.spec.SecretKeySpec(tripleDESKey, 0, 24, "3DES");

            if (secretKey != null)
            {
                return secretKey;
            }
            else
            {
                throw new com.ibm.websphere.crypto.KeyException ("Key could not be generated.");
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();  // handle exception
        }
    }
}
```

If you are developing a key pair, refer to the following example, which uses the KeyPairGenerator class to implement the com.ibm.websphere.crypto.KeyPairGenerator interface.

```
package com.ibm.test;

import java.util.*;
import javax.crypto.spec.SecretKeySpec;
import com.ibm.websphere.crypto.KeyException;

/**
 * This implementation defines the method to generate a java.security.KeyPair.
 * When a keyGeneration class implements this method, the generateKeyPair method
 * is called and a KeyPair is stored in the keystore. The isKeyPair
 * attribute is ignored since the KeyGenerationClass is an
 * implementation of KeyPairGenerator. The isKeyPair attributes is for when
 * the keys already exist in a KeyStore, and are just read (not
 * generating them).
 *
 * @author IBM Corporation
 * @version WebSphere Application Server 6.1
 * @since WebSphere Application Server 6.1
 **/
public class KeyPairGenerator implements com.ibm.websphere.crypto.KeyPairGenerator
{
    private java.util.Properties customProperties = null;

    public KeyPairGenerator()
    {
    }

     /**
      * This method is called to pass any custom properties configured with
      * the KeySet to the implementation of this interface.
     *
      * @param java.util.Properties
      **/
    public void init (java.util.Properties customProps)
    {
        customProperties = customProps;
    }
```

```
     /**
     * This method is called whenever a key needs to be generated either
      * from the schedule or manually requested and isKeyPair=true in the KeySet
      * configuration. The key is stored in the KeyStore referenced by
      * the configured KeySet which contains the keyGenerationClass implementing
      * this interface. The implementation of this interface manages the
      * type of the key. The user of the KeySet must know the type that
      * is returned by this keyGenerationClass.
      *
      * @return com.ibm.websphere.crypto.KeyPair
      * @throws com.ibm.websphere.crypto.KeyException
      **/
    public com.ibm.websphere.crypto.KeyPair generateKeyPair () throws KeyException
    {
        try
        {
            java.security.KeyPair keyPair = generateKeyPair();

            // Store as SecretKeySpec
            if (keyPair != null)
            {
                java.security.PrivateKey privKey = keyPair.getPrivate();
                java.security.PublicKey pubKey = keyPair.getPublic();

                SecretKeySpec publicKeyAsSecretKey = new SecretKeySpec
                    (pubKey.getEncoded(), "RSA_PUBLIC");
                SecretKeySpec privateKeyAsSecretKey = new SecretKeySpec
                    (privKey.getEncoded(), "RSA_PRIVATE");

                com.ibm.websphere.crypto.KeyPair pair = new
    com.ibm.websphere.crypto.KeyPair(publicKeyAsSecretKey, privateKeyAsSecretKey);
                return pair;
            }
            else
            {
                throw new com.ibm.websphere.crypto.KeyException ("Key pair could
                not be generated.");
            }
        }
        catch (Exception e)
        {
            e.printStackTrace();  // handle exception
        }
    }
}
```

This interface returns a com.ibm.websphere.crypto.KeyPair key pair, which can contain either a X509Certificate and PrivateKey object or PublicKey and PrivateKey objects. If the com.ibm.websphere.crypto.KeyPair interface contains aX509Certificate and PrivateKey object, the certificate and private key are stored in the keystore. Consequently, they can use any KeyStore type.

If the com.ibm.websphere.crypto.KeyPair interface contains PublicKey and PrivateKey objects, you must convert the encoded values to the SecretKeySpec object in order to store them. The WebSphere Application Server runtime stores and retrieves the key pair as secret keys. The runtime converts the key pair back to PublicKey and PrivateKey objects when the server retrieves the pair during the handshake.

Use the following constructors to develop the com.ibm.websphere.crypto.KeyPair interface:

- Public and private constructor

```
public KeyPair(java.security.Key publicKey, java.security.Key privateKey)
```

- Certificate and private constructor.

```
public KeyPair(java.security.cert.Certificate[] certChain,
java.security.Key privateKey)
```

The previous example code shows the KeyPairGenerator class using the public and private constructor. Each call to this class generates a new and unique key pair, and this class is invoked by a KeySet to create a new key pair when `isKeyPair=true`. The version number in the key set increments each time it is called.

Key set groups collection

Use this page to manage groups of public, private, and shared keys. These key groups enable the application server to control multiple sets of Lightweight Third Party Authentication (LTPA) keys.

To view this administrative console page, click **Security > SSL certificate and key management**. Under Configuration settings, click **Manage endpoint security configurations > {Inbound | Outbound} > *ssl_configuration***. Under Related items, click **Key set groups**.

Table 165. Key set groups buttons. This table lists the key set groups buttons.

Button	Resulting action
New	Adds a key set group. A key set group combines one or more key sets together as a single key set group. It allows the generation of multiple different types of keys to occur at the same time. A single key set represents one type of key, so a key set group allows you to group the different types.
Delete	Deletes an existing key set group. You must be sure that there are no other references to this key set group before you delete it.
Generate keys	Generates keys for key set group. The system generates keys for each key set within the key set group so that the keys remain synchronized with each other in terms of version. You must configure a valid key generation class and a key store that is writable. See the com.ibm.websphere.crypto.KeySetHelper application programming interfaces (APIs) to enable the use of keys that are managed by a KeySetGroup or KeySet.

Key set group name:

Specifies the name of the key set group used to reference it.

Automatically generate keys:

Specifies that the keys are to be generated automatically on a schedule.

Key set groups settings

Use this page to create new key set groups.

To view this administrative console page, click **Security** > **SSL certificate and key management**. Under Configuration settings, click**Manage endpoint security configurations** > **{Inbound | Outbound}** > ***ssl_configuration***. Under Related items, click **Key set groups** > **New**.

Key set group name:

Specifies the name of key set group used. This name can be referenced using the com.ibm.websphere.crypto.KeySetHelper API to retrieve the managed keys from an application.

Data type: Text

Management scope:

Specifies the scope where this Secure Sockets Layer (SSL) configuration is visible. For example, if you choose a specific node, then the configuration is only visible on that node and any servers that are part of that node.

Data type List
Range: Applicable scopes

Key sets:

Specifies a set of key instances of the same type for use in cryptographic operations.

This setting has the following options:

Add Specifies to add the selected key set part of this key set group.

Remove
Specifies to remove the selection from the **Key sets** list.

Automatically generate keys:

Specifies that the keys are generated automatically on a schedule. When a new key is generated, the `security.xml` is updated and saved by the runtime to track the key reference version. This can cause save conflicts when updating the same file from admin applications.

gotcha: Starting with Versions 6.1.0.23 and 7.0.0.3, the default value for this property is Disabled.

If you try to enable this property, and automatic synchronization is off in any node, the following administrative console message displays:

```
Warning: At least one node in the cell was unreachable or is not configured to automatically synchronize.
It is strongly recommended that you verify your node settings, and do not enable automatic
generation of LTPA keys while automatic synchronization is disabled on any node.
```

Default for Versions 7.0, and 7.0.0.1:	Enabled
Default for Versions 7.0.0.3 and higher:	Disabled

Scheduled time for generation:

Specifies the scheduled time when the system generates selected key set group or groups. You can specify the scheduled time in hours and minutes; specify either A.M. or P.M., or specify 24-hour. You can also specify the day of the week you want the scheduled event to occur. It is recommended that you set this event to occur during a low peak time, especially for keys that are used by runtime for token validation.

Data type	Integer
Default:	8, 0 A.M.
Range:	1–12, with a A.M. or P.M. setting 0–59, with a 24-hour setting

Generate on a specific day:

Specifies whether to have the generation occur on a specific day of the week. It is best to auto-generate keys during a low peak day.

This setting has the following options:

Weekday

Specifies the day of the week on which the expiration monitor will run if the Check on a specific day option is selected.

Default:	Sunday
Range:	Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday

Repeat interval

Specifies the period of time, in weeks, between each schedule time to check for expired certificates or the interval between schedule checks.

Default:	4 weeks

Generate at an interval:

Specifies to generate keys at the specified frequency regardless of the day of the week on which generation occurs.

Default: Disabled

This setting has the following options:

Repeat interval
Specifies the period of time, in days, between each schedule time to check for expired certificates or the interval between schedule checks.

Default: 7 days

Next start date:

Specifies the date for the next scheduled check. This allows the deployment manager to be stopped and restarted without resetting the date.

Auditing the security infrastructure

You can use the Auditing Facility to report and track auditable events to ensure the integrity of your system.

Before you begin

Before enabling the security auditing subsystem, you must enable global security in your environment.

About this task

Note: The security auditing subsystem has been introduced as part of the security infrastructure. The primary responsibility of the security infrastructure is to prevent unauthorized access and usage of resources. Utilizing security auditing has two primary goals:

- Confirming the effectiveness and integrity of the existing security configuration.
- Identifying areas where improvement to the security configuration might be needed.

Security auditing achieves these goals by providing the infrastructure that allows you to implement your code to capture and store supported auditable security events. During run time, all code other than the Java EE 5 application code is considered to be trusted. Each time a Java EE 5 application accesses a secured resource, any internal application server process with an audit point included can be recorded as an auditable event.

The security auditing subsystem has the ability to capture the following types of auditable events:

- Authentication
- Authorization
- Principal/Credential Mapping
- Audit policy management
- Delegation

Restriction: Audit instrumentation has not been included in the web services client run time.

These types of events can be recorded into audit log files. Each audit log has the option to be signed and encrypted to ensure data integrity. These audit log files can be analyzed to discover breaches over the existing security mechanisms and to discover potential weaknesses in the current security infrastructure. Security event audit records are also useful for providing evidence of accountability and nonrepudiation as well as vulnerability analysis. The security auditing configuration provides four default filters, a default audit service provider, and a default event factory. The default implementation write to a binary text-file based log. Use this topic to customize your security auditing subsystem.

Procedure

1. “Enabling the security auditing subsystem”

   Security auditing will not be performed unless the audit security subsystem has been enabled. Global security must be enabled for the security audit subsystem to function, as no security auditing occurs if global security is not also enabled.

2. Assign the auditor role to a user

   A user with the auditor role is required to enable and configure the security auditing subsystem. It is important to require strict access control for security policy management. The auditor role has been created providing granularity to allow for separation of the auditing role from the authority of the administrator. When Security Auditing is initially enabled, the cell administrator has auditor privileges. If the environment requires separation of privileges, then changes will need to be made to the default role assignments.

3. “Creating security auditing event type filters” on page 1853

   You can configure event type filters to only record a specific subset of auditable event types in your audit logs. Filtering the event types that are recorded makes for easier analysis of your audit records by ensuring only those records important to your environment are archived.

4. Configuring the audit service provider.

   The audit service provider is used to format the audit data object that was passed to it before outputting the data to a repository. A default audit service provider implementation is in included. See “Configuring the default audit service providers for security auditing” on page 1864 for more details on the default implementation. A third party implementation can also be coded and used. See “Configuring a third party audit service providers for security auditing” on page 1868 for more details on this implementation.

5. “Configuring audit event factories for security auditing” on page 1869

   The audit event factory gathers the data associated with the auditable events and creates an audit data object. The audit data object is then sent to the audit service provider to be formatted and recorded to the repository.

6. “Protecting your security audit data” on page 1872

   It is important to secure and ensure the data integrity of the recorded audit data. To ensure that access to the data is restricted and tamper proof, you can encrypt and sign your audit data.

7. “Configuring security audit subsystem failure notifications” on page 1861

   Notifications can be enabled to generate alerts when the security auditing subsystem experiences a failure. Notifications can be configured to record an alert in the System logs or can be configured to send an alert through email to a specified list of recipients.

Results

After successfully completing this task, you audit data will be recorded for the selected auditable events that were specified in the configuration.

What to do next

After configuring security auditing, you can analyze your audit data for potential weaknesses in the current security infrastructure and to discover security breaches that may have occurred over the existing security mechanisms. You can also use the security auditing subsystem to provide data for problem determination. If the default audit service provider was selected, the resulting binary audit log file can be read using the Audit Reader.

Enabling the security auditing subsystem

Security auditing will not be performed unless the audit security subsystem has been enabled. Global security must be enabled for the security audit subsystem to function, as no security auditing occurs if global security is not also enabled.

Before you begin

Before enabling security auditing subsystem, enable global security in your environment.

About this task

The recording of auditable security events is achieved by enabled the security auditing subsystem. Follow these steps to enable the security auditing subsystem.

Procedure

1. Click **Security > Security auditing**.
2. Select Enable security auditing. The Enable security auditing check box is not selected by default. This check box must be selected to allow security auditing to be performed with the configurations that have been specified in the `audit.xml` file.

   Note: The `audit.xml` file is used to store the audit subsystem configurations. Changes to the security auditing subsystem should be made with the user interface or the wsadmin utility. This file should not be edited manually.
3. Select the action from the Audit subsystem failure action dropdown menu to be perform when an audit subsystem failure occurs. Notifications configured to warn of a security auditing subsystem failure will not be posted if the No Warning option is selected for this field. If you select either the Log warning or the Terminate server option, then you must also configure a notification for the action to be performed.
4. Select the Auditor ID from the dropdown menu. The auditor role is needed to make changed to the security auditing configurations. By default, when auditing is first enabled, the primary administrator is also given the auditor role. The primary administrator can then add the auditor role to other users. After the auditor role is added to other users, the auditor role can be removed from the administrator to create a separation of authority between the auditor and the administrator. The Auditor ID is the user considered to be the primary auditor.
5. Optional: Select Enable verbose auditing. When an auditable event is recorded, a default set of audit data is included in the audit data object and recorded to the repository. An additional set of audit data is made available by enabling verbose auditing.
6. Click **Apply**.
7. Restart the application server. The application server must be restarted before the changes go into effect.

Results

The successful competition of these steps results in the security auditing subsystem being enabled.

What to do next

After enabling the security auditing subsystem, refinements can be made to the configuration. You might want to modify the access control of the audit subsystem to separate the authority of the administrator from the authority of the auditor. If no changes to your access control are needed, then you can configure the types of auditable security events should be recorded. To configure the types of events that are recorded, click **Event type filters**.

Security Auditing detail

The Security auditing subsystem can be enabled and configured from this panel, by users assigned the auditor role.

To view this administrative console page, click **Security > Security Auditing**. If Enable security auditing is not selected, then all of the other fields on this panel will be disabled.

Enable security auditing:

The Enable security auditing check box allows users to enable or disable Security Auditing. By default, Security Auditing will not be enabled. This field corresponds with the auditEnabled field in the `audit.xml` file.

Audit subsystem failure action:

The Audit subsystem failure action setting describes the behavior of the application server in the event of a failure in the auditing subsystem. Audit Notifications must be configured in order for notifications of a failure in the audit subsystem to be logged. If security auditing is not enabled, then these actions will not be performed. Failures can include an error in the interface or in the event processing. By default, the audit subsystem failure action setting is set to No warning.

The Audit subsystem failure action dropdown menu has the following options:

- No warning

  The No warning action specifies that the auditor will not be notified of a failure in the audit subsystem. The product will continue processing but audit reporting will be disabled.
- Log warning

  The Log warning action specifies that the auditor will be notified of a failure in the audit subsystem. The product will continue processing but audit reporting will be disabled.
- Terminate server

  The Terminate server action specifies the application server to gracefully quiesce when an unrecoverable error occurs in the auditing subsystem. If email notifications are configured, the auditor will be sent a notification that an error has occurred. If logging to the system log is configured, the notification of the failure will be logged to the system file.

Primary auditor user name:

The Primary auditor user name dropdown menu defines a valid user which exists in the current user registry and for whom the auditor role has been given. By default, this field is blank and is a required field.

Enable verbose auditing:

The Enable verbose auditing option determines the amount of audit data that is reported in an audit record. Verbose mode captures all the auditable data points, whereas not enabling verbose mode captures only a subset of the available data. This option is disabled by default.

Context object fields

Each auditable event has an associated set of information that is available for logging. This information is grouped into specific context objects. The context objects that are available for logging a specific event are specified by the event type. This topic details the information that exists for each context object and specifies whether the information is logged by default or is only logged when the verbose logging option is enabled.

The SessionContextObj object

Table 166. SessionContextObj fields. This table lists the SessionContextObj fields.

Field	Type	Description	Default or Verbose logging
sessionId	String	An identifier for the user session	Default
remoteAddr	String	The IP address for the remote host	Default
remotePort	String	The port of the remote host	Default
remoteHost	String	The host name of the remote host	Default

The PropagationContextObj object

Table 167. PropagationContextObj fields. This table lists the PropagationContextObj fields.

Field	Type	Description	Default or Verbose logging
firstCaller	String	The identity of the first user in the caller list	Default
callerList	String array	A list of names representing the identities of the users	Verbose

The RegistryContextObj object

Table 168. RegistryContextObj fields. This table lists the RegistryContextObj fields.

Field	Type	Description	Default or Verbose logging
type	String	The type of user registry being used, such as LDAP or AIX	Default

The ProcessContextObj object

Table 169. ProcessContextObj fields. This table lists the ProcessContextObj fields.

Field	Type	Description	Default or Verbose logging
domain	String	The domain to which the user belongs	Verbose
realm	String	The registry partition to which the user belongs	Default

The EventContextObj object

Table 170. EventContextObj fields. This table lists the EventContextObj fields.

Field	Type	Description	Default or Verbose logging
lastEventTrailId	String	The last ID associated with a given transaction	Verbose
eventTrailId	String array	An array of IDs that allow events that belong to a given transaction to be correlated	Default
creationTime	Date	The date an event was created	Default
globalInstanceId	Long	The unique identifier of this event	Default

The DelegationContextObj object

Table 171. DelegationContextObj fields. This table lists the DelegationContextObj fields.

Field	Type	Description	Default or Verbose logging
delegationType	String	no delegation, simple delegation, method delegation or switch user delegation	Default
roleName	String	The Run as role being used: runAsClient, runAsSpecified, runAsSystem, own ID	Default
identityName	String	Information about the mapped user	Default

The AuthnContextObj object

Table 172. AuthnContextObj fields. This table lists the AuthnContextObj fields.

Field	Type	Description	Default or Verbose logging
authnType	String	The type of authentication used	Default

The ProviderContextObj object

Table 173. ProviderContextObj fields. This table lists the ProviderContextObj fields.

Field	Type	Description	Default or Verbose logging
provider	String	The provider of the authentication or authorization service	Default
providerStatus	String	Status of whether the authentication or authorization event processed successfully by the provider	Default

The AuthnMappingContextObj object

Table 174. AuthnMappingContextObj fields. This table lists the AuthnMappingContextObj fields.

Field	Type	Description	Default or Verbose logging
mappedSecurityDomain	String	The security domain after mapping has occurred	Default
mappedRealm	String	The realm after mapping has occurred	Default
mappedUserName	String	The user name after mapping has occurred	Default

The AuthnTermContextObj object

Table 175. AuthnTermContextObj fields. This table lists the AuthnTermContextObj fields.

Field	Type	Description	Default or Verbose logging
terminateReason	String	The reason authentication ended	Default

The AccessContextObj object

Table 176. AccessContextObj fields. This table lists the AccessContextObj fields.

Field	Type	Description	Default or Verbose logging
progName	String	The name of the program that was involved in the event	Default
action	String	The action being performed.	Default
registryUserName	String	The name of the user in the registry	Default
appUserName	String	The name of the user within an application	Default
accessDecision	String	The decision of the authorization call	Default
resourceName	String	The name of the resource in the context of the application	Default
resourceType	String	The type of resource	Default
resourceUniqueId	Long	The unique identifier of the resource	Default
permissionsChecked	String array	The permissions that were checked during the authorization call	Default
permissionsGranted	String array	The permissions that were granted during the authorization call	Default
rolesChecked	String array	The roles that were checked during the authorization call	Default
rolesGranted	String array	The roles that were granted during the authorization call	Default

The PolicyContextObj object

Table 177. PolicyContextObj fields. This table lists the PolicyContextObj fields.

Field	Type	Description	Default or Verbose logging
policyName	String	The name of the policy	Default

Table 177. PolicyContextObj fields (continued). This table lists the PolicyContextObj fields.

Field	Type	Description	Default or Verbose logging
policyType	String	The type of policy	Default

The KeyContextObj object

Table 178. KeyContextObj fields. This table lists the KeyContextObj fields.

Field	Type	Description	Default or Verbose logging
keyLabel	String	The key or certificate label	Default
keyLocation	String	The physical location of the key database	Default
certLifetime	Date	The date when a certificate expires	Default

The CipherContextObj object

Table 179. CipherContextObj fields. This table lists the CipherContextObj fields.

Field	Type	Description	Default or Verbose logging
cipherData	Byte array	The cipher data that is captured	Verbose

The MgmtContextObj object

Table 180. MgmtContextObj fields. This table lists the MgmtContextObj fields.

Field	Type	Description	Default or Verbose logging
mgmtType	String	The type of management operation	Default
mgmtCommand	String	The application-specific command that was performed	Default
targetInfoAttributes	Target Atrribute array	Information about one or more secondary objects involved in this operation	Verbose

The ResponseContextObj object

Table 181. ResponseContextObj fields. This table lists the ResponseContextObj fields.

Field	Type	Description	Default or Verbose logging
url	String	The URL of the HTTP request	Default
httpRequestHeaders	Attributes array	The HTTP request headers provided by the client	Verbose
httpResponseHeaders	Attributes array	The HTTP response headers returned by the server	Verbose

The CustomPropertyContextObj object

Table 182. CustomPropertyContextObj fields. This table lists the CustomPropertyContextObj fields.

Field	Type	Description	Default or Verbose logging
key	String	The label representing the custom property key name	Verbose
value	Object	The object value of the custom property	Verbose

Creating security auditing event type filters

Event type filters are used to specify the types of auditable security events that are audited. Default event type filters are included with the product, but you can also configure new event type filters to specify a subset of auditable event types to be recorded by the security auditing subsystem.

Before you begin

Before configuring security auditing filters and the rest of the security auditing subsystem, enable global security in your environment. You must be assigned the auditor role to complete this task. Event type filters are used to specify what events are audited. The amount of data that is recorded for each event is specified with the **Enable verbose auditing** check box on the same panel used to enable the auditing subsystem. Navigate to **Security** > **Security auditing** to enable security auditing and determine the data recorded for each event.

About this task

Table 183. Commonly used event type filters by default in the audit.xml template file. The application server provides the following commonly used event type filters by default in the `audit.xml` template file:

Name	Event name	Outcome of event
DefaultAuditSpecification_1	SECURITY_AUTHN	SUCCESS
DefaultAuditSpecification_2	SECURITY_AUTHN	DENIED
DefaultAuditSpecification_3	SECURITY_RESOURCE_ACCESS	SUCCESS
DefaultAuditSpecification_4	SECURITY_AUTHN	REDIRECT

New event type filters can be created, or the existing default filters can be extended, to capture more event types and outcomes. Use this task to create new event type filters.

.

Procedure

1. Click **Security > Security Auditing > Event type filters> New**.
2. Enter the unique name that should be associated with this event type filter configuration in the Name field.
3. Specify the events that should be recorded when this filter is applied:
   a. Select the events that you want to be audited from the Selectable events list.
   b. Click **Add >>** to add the selected events to the Enabled events list.
   c. Select the outcomes that you want to be audited from the Selectable event outcomes list.
   d. Click **Add >>** to add the selected outcomes to the Enabled event outcomes lists.
4. Click **OK**.

Results

The successful completion of this task results in the creation of an event type filter than can be selected by the audit service providers and audit event factories to gather and record a specific set of auditable security events.

What to do next

After creating an event type filter, the filter must be specified in the audit service provider and the audit event factory to be used to gather or report audit data. The next step in configuring the security auditing subsystem is you should configure an audit service provider to define where the audit data will be archived.

Auditable security events

Auditable security events are security events that have audit instrumentation added to the security run time code to enable them to be recorded. Event filters are configured to specify which auditable security events are recorded to the audit log files.

The following list describes each valid auditable event that you can specify as an enabled event type when creating an event filter:

Table 184. Event types. Valid auditable events can be specified as an enabled event type when creating an event filter:

Event name	Description
SECURITY_AUTHN	Audits all authentication events
SECURITY_AUTHN_MAPPING	Audits events that record mapping of credentials where two user identities are involved
SECURITY_AUTHN_TERMINATE	Audits authentication termination events such as a timeout, terminated session, or user-initiated logging out
SECURITY_AUTHZ	Audits events related to authorization checks when the system enforces access control policies
SECURITY_RUNTIME	Audits runtime events such as the starting and the stopping of security servers. This event type is not meant for administrative operations performed by a system administrator as such operations need to use the other SECURITY_MGMT_* event types.
SECURITY_MGMT_AUDIT	Audits events that record operations related to the audit subsystem such as starting audit, stopping audit, turning audit on or off, changing configuration of audit filters or level, archiving audit data, purging audit data, and so on.
SECURITY_RESOURCE_ACCESS	Audits events that record all accesses to a resource. Examples are all accesses to a file, all HTTP requests and responses to a given web page, and all accesses to a critical database table
SECURITY_SIGNING	Audits events that record signing such as signing operations used to validate parts of a SOAP Message for web services
SECURITY_ENCRYPTION	Audits events that record encryption information such as encryption for web services
SECURITY_AUTHN_DELEGATION	Audits events that record delegation, including identity assertion, RunAs, and low assertion. Used when the client identity is propagated or when delegation involves the use of a special identity. This event type is also used when switching user identities within a given session.
SECURITY_AUTHN_CREDS_MODIFY	Audits events to modify credentials for a given user identity

For each audit event type, you must specify an outcome. Valid outcomes include SUCCESS, FAILURE, REDIRECT, ERROR, DENIED, WARNING, and INFO. Not all outcomes are applicable with all event types.

Note: Support for the SECURITY_RUNTIME auditing event type has been fully implemented for this release of WebSphere Application Server. It audits runtime events such as the starting and the stopping of security servers.

Event type filter settings

The Event type filter settings panel is used by an auditor to manage and create event type filters. Default event type filters have been included, this panel allows additional event type filters to be added. Existing event type filters are also managed using this panel.

To view this administrative console page, click one of the following paths:

- **Security > Security Auditing > Event type filters >** *event_type_filter_name*.
- **Security > Security Auditing > Event type filters > New** .

Name:

The Name field specifies the unique name of the event type filter.

Enabled:

The state of enablement of the filter is defined by the Enable check box. This field is represented as a boolean value. A value of true specifies that the enable field associated with the audit specification in the `audit.xml` is set to true. It does not imply that all configured event factories and service providers will be using this filter.

Filters still need to be configured for each event factory and service provider. Filters are enabled by default during configuration. However, if a filter has the enabled checkbox set to false, the filter will not gather or report data for the events and outcomes defined in that filter.

Events to associate with an audit filter:

The Events to associate with an audit filter field specifies the auditable security events to be associated with this filter.

- Selectable events:

  The Selectable events list displays the available auditable security events. To enable an event for this filter, select the event from the Selectable event outcomes list and then click **Add**.

- Enabled events:

  The Enabled events list displays the audit security events that are currently enabled for this filter. To disable an event for this filter, select the event from the Enabled events list and then click **Remove**.

Event outcomes to associate with an audit filter:

The Event outcomes to associate with an audit filter field specifies the auditable security event outcomes to be associated with this filter.

- Selectable event outcomes:

  The Selectable event outcomes list displays the available auditable security event outcomes. To enable an event outcome for this filter, select the event outcome from the Selectable event outcomes list and then click **Add**.

- Enabled event outcomes:

  The Enabled event outcomes list displays the audit security event outcomes that are currently enabled for this filter. To disable an event outcome for this filter, select the event outcome from the Enabled event outcomes list and then click **Remove**.

Event type filters collection

The Event type filters panel displays a listing of all configured audit specifications with their unique names, the state of their enablement, and the event types and event outcomes that are specified for each configuration.

To view this administrative console page, click **Security > Security Auditing > Event type filters**.

Name:

The Name field displays the unique name of the event type filter that is being represented.

Enable:

The Enable check box species the state of enablement for the filter. This field is represented as a boolean value. A value of true specifies that the enable field associated with the audit specification in the `audit.xml` is set to true. It does not imply that all configured event factories and service providers will be using this filter. Filters are enabled by default when they are created. Even though it is enabled by default when it is created, the event type filter must be specified for the event factory and the audit service provider before it is actually used,

Filters still need to be configured for each event factory and service provider. A filter that is configured for an event factory or a service provider that has Enabled set to false, will not gather or report data for the events and outcomes defined in that filter.

Events and outcomes:

The event types and the event outcomes that are specified by this filter. The specifications are listed in the form event_type:event_outcome and separated by commas if multiple combinations are specified by the event type filter.

Example: Generic Event Interface

This interface is used for processing generic audit events. Other interfaces can be defined which extend this interface to process specific audit event groupings, such as security events, transaction events, or other custom groupings. For WebSphere Application Server version 7.0, only security types of events are supported.

Generic Event Interface

Specific implementations might be developed to handle the data in a particular internal format. When the buildEvent() method is called, the implementation must then build the specified base event type using the internal information it has stored. After the information has been stored into a GenericEvent instance, the GenericEvent interface provides a generic way of handling the event.

```
public interface GenericEvent {

  /** * Property name used to specify the base event type to the
 * {@link GenericEvent#buildEvent} method.
 */
public static final String BASE_EVENT_TYPE = GenericEvent.class.getName() + ".baseEventType";

/**
* Returns the eventType of the event. The eventType distinguishes between these
* related events.
* The eventType depends on the particular implementation
* of the GenericEvent. For example, the Security Event implmentation has
* eventTypes such as SECURITY_AUTHN and SECURITY_AUTHZ.
* @return eventType - the eventType of the event
*/

public String getEventType();
/**
* Returns the creationTime, the creation time of the event.
* @return creationTime - the creation time of the event
*/ public Date getCreationTime(); /** * Returns the version, the version of the event.
* @return version - the version of the event
*/

public String getVersion (Properties props) throws GenericEventConfigurationException;
/**
* Returns the globalInstanceId, which is a globally unique instance
* identifier for the event.
* @return globalInstanceId - a globally unique instance identifier for the event
*/

public Long getGlobalInstanceId();
/**
* Verifies whether the event is valid; which depends on the particular
* implementation of the GenericEvent. If the event is not valid, an
* GenericEventValidationException error occurs.
*/

public void validate() throws GenericEventValidationException;
 /**
* Returns the internally wrapped base event instance after
* completing and validating the current instance of the GenericEvent.
* An GenericEvent implementation can maintain its information
* in any undisclosed internal format. The buildEvent()
* method that specifies that a specific base event type be built
* using the internal information. This allows GenericEvent implementations
* to support multiple base event formats. Thus the GenericEvent implmentation
* provides a layer of abstraction higher than the base event type.
* @param properties The value of the property BASE_EVENT_TYPE
* defines the type of the base event * @return the internally wrapped base event instance
* @throws GenericEventConfigurationException if the base event type is invalid
* or the JAR files to support that event type are not available.
* @throws GenericEventCompletionException if event completion has failed.
* @throws GenericEventValidationException if the validation has failed. This is
* validation as is performed by the validate() method.
*/

public Object buildEvent(Properties properties)
  throws GenericEventConfigurationException,
      GenericEventValidationException,
       GenericEventCompletionException;
/**
* Returns the wrapped base event instance as a string after
* completing and validating the current instance of the GenericEvent.
* An GenericEvent implementation can maintain its information
* in any undisclosed internal format. It is the buildEventString()
* method that specifies that a specific base event type be built
* using the internal information. This allows GenericEvent implementations
* to support multiple base event formats. Thus the GenericEvent implmentation
* provides a layer of abstraction higher than the base event type.
```

```
* @param properties The value of the property BASE_EVENT_TYPE
* defines the type of the base event
* @return the wrapped base event instance as a String
* @throws GenericEventConfigurationException if the base event type is invalid
* or the JAR files to support that event type are not available.
* @throws GenericEventCompletionException if event completion has failed.
* @throws GenericEventValidationException if the validation has failed. This is
* validation as is performed by the validate() method.
*/

public String buildEventString(Properties properties)
  throws GenericEventConfigurationException,
      GenericEventValidationException,
      GenericEventCompletionException;
}
```

Context objects for security auditing

Each event has an associated set of information that is available for logging. This information is grouped into specific context objects. The context objects that are available for logging a specific event are specified by the event type. All event types have the sessionContextObj, eventContextObj, accessContextObj, propogationContextObj, processContextObj and registryContextObj objects. This topic specifies which additional context objects are available for each event type.

Table 185. Context objects associated with event types. The following table describes the context objects associated with event types.

Event Type	Additional Context Objects
SECURITY_AUTHN	authnContextObj, providerContextObj
SECURITY_AUTHN_DELEGATION	delegationContextObj
SECURITY_AUTHN_MAPPING	authnMappingContextObj, providerContextObj
SECURITY_AUTHZ	providerContextObj, policyContextObj
SECURITY_ENCRYPTION	keyContextObj
SECURITY_MGMT_AUDIT	mgmtContextObj
SECURITY_RESOURCE_ACCESS	responseContextObj

For more details on the auditable data that is gather for each of these context objects, see the information for context object fields.

Context object fields

Each auditable event has an associated set of information that is available for logging. This information is grouped into specific context objects. The context objects that are available for logging a specific event are specified by the event type. This topic details the information that exists for each context object and specifies whether the information is logged by default or is only logged when the verbose logging option is enabled.

The SessionContextObj object

Table 186. SessionContextObj fields. This table lists the SessionContextObj fields.

Field	Type	Description	Default or Verbose logging
sessionId	String	An identifier for the user session	Default
remoteAddr	String	The IP address for the remote host	Default
remotePort	String	The port of the remote host	Default
remoteHost	String	The host name of the remote host	Default

The PropagationContextObj object

Table 187. PropagationContextObj fields. This table lists the PropagationContextObj fields.

Field	Type	Description	Default or Verbose logging
firstCaller	String	The identity of the first user in the caller list	Default
callerList	String array	A list of names representing the identities of the users	Verbose

The RegistryContextObj object

Table 188. RegistryContextObj fields. This table lists the RegistryContextObj fields.

Field	Type	Description	Default or Verbose logging
type	String	The type of user registry being used, such as LDAP or AIX	Default

The ProcessContextObj object

Table 189. ProcessContextObj fields. This table lists the ProcessContextObj fields.

Field	Type	Description	Default or Verbose logging
domain	String	The domain to which the user belongs	Verbose
realm	String	The registry partition to which the user belongs	Default

The EventContextObj object

Table 190. EventContextObj fields. This table lists the EventContextObj fields.

Field	Type	Description	Default or Verbose logging
lastEventTrailId	String	The last ID associated with a given transaction	Verbose
eventTrailId	String array	An array of IDs that allow events that belong to a given transaction to be correlated	Default
creationTime	Date	The date an event was created	Default
globalInstanceId	Long	The unique identifier of this event	Default

The DelegationContextObj object

Table 191. DelegationContextObj fields. This table lists the DelegationContextObj fields.

Field	Type	Description	Default or Verbose logging
delegationType	String	no delegation, simple delegation, method delegation or switch user delegation	Default
roleName	String	The Run as role being used: runAsClient, runAsSpecified, runAsSystem, own ID	Default
identityName	String	Information about the mapped user	Default

The AuthnContextObj object

Table 192. AuthnContextObj fields. This table lists the AuthnContextObj fields.

Field	Type	Description	Default or Verbose logging
authnType	String	The type of authentication used	Default

The ProviderContextObj object

Table 193. ProviderContextObj fields. This table lists the ProviderContextObj fields.

Field	Type	Description	Default or Verbose logging
provider	String	The provider of the authentication or authorization service	Default
providerStatus	String	Status of whether the authentication or authorization event processed successfully by the provider	Default

The AuthnMappingContextObj object

Table 194. AuthnMappingContextObj fields. This table lists the AuthnMappingContextObj fields.

Field	Type	Description	Default or Verbose logging
mappedSecurityDomain	String	The security domain after mapping has occurred	Default
mappedRealm	String	The realm after mapping has occurred	Default
mappedUserName	String	The user name after mapping has occurred	Default

The AuthnTermContextObj object

Table 195. AuthnTermContextObj fields. This table lists the AuthnTermContextObj fields.

Field	Type	Description	Default or Verbose logging
terminateReason	String	The reason authentication ended	Default

The AccessContextObj object

Table 196. AccessContextObj fields. This table lists the AccessContextObj fields.

Field	Type	Description	Default or Verbose logging
progName	String	The name of the program that was involved in the event	Default
action	String	The action being performed.	Default
registryUserName	String	The name of the user in the registry	Default
appUserName	String	The name of the user within an application	Default
accessDecision	String	The decision of the authorization call	Default
resourceName	String	The name of the resource in the context of the application	Default
resourceType	String	The type of resource	Default
resourceUniqueId	Long	The unique identifier of the resource	Default
permissionsChecked	String array	The permissions that were checked during the authorization call	Default
permissionsGranted	String array	The permissions that were granted during the authorization call	Default
rolesChecked	String array	The roles that were checked during the authorization call	Default
rolesGranted	String array	The roles that were granted during the authorization call	Default

The PolicyContextObj object

Table 197. PolicyContextObj fields. This table lists the PolicyContextObj fields.

Field	Type	Description	Default or Verbose logging
policyName	String	The name of the policy	Default
policyType	String	The type of policy	Default

The KeyContextObj object

Table 198. KeyContextObj fields. This table lists the KeyContextObj fields.

Field	Type	Description	Default or Verbose logging
keyLabel	String	The key or certificate label	Default
keyLocation	String	The physical location of the key database	Default
certLifetime	Date	The date when a certificate expires	Default

The CipherContextObj object

Table 199. CipherContextObj fields. This table lists the CipherContextObj fields.

Field	Type	Description	Default or Verbose logging
cipherData	Byte array	The cipher data that is captured	Verbose

The MgmtContextObj object

Table 200. MgmtContextObj fields. This table lists the MgmtContextObj fields.

Field	Type	Description	Default or Verbose logging
mgmtType	String	The type of management operation	Default
mgmtCommand	String	The application-specific command that was performed	Default
targetInfoAttributes	Target Atrribute array	Information about one or more secondary objects involved in this operation	Verbose

The ResponseContextObj object

Table 201. ResponseContextObj fields. This table lists the ResponseContextObj fields.

Field	Type	Description	Default or Verbose logging
url	String	The URL of the HTTP request	Default
httpRequestHeaders	Attributes array	The HTTP request headers provided by the client	Verbose
httpResponseHeaders	Attributes array	The HTTP response headers returned by the server	Verbose

The CustomPropertyContextObj object

Table 202. CustomPropertyContextObj fields. This table lists the CustomPropertyContextObj fields.

Field	Type	Description	Default or Verbose logging
key	String	The label representing the custom property key name	Verbose
value	Object	The object value of the custom property	Verbose

Configuring security audit subsystem failure notifications

Notifications can be generated by a failure of the security audit subsystem. The security audit subsystem notifications can alert auditors that the security audit system is no longer recording auditable security events. Notifications are generated by a failure of the auditing subsystem, they are not related to any auditable security events or event outcome that has occurred. Notifications triggered by an event or an event outcome are not supported.

Before you begin

Before configuring notifications, enable global security and the security audit subsystem in your environment. You must be assigned the auditor role to complete this task.

About this task

If a problem is experienced with the security audit subsystem, then a notification can be generated. This is an alert that security events are no longer being audited. Notification can be written to the system log file or can be sent to a specified group of users as an email. You are able to configure notifications to alert the auditor of a problem using both of these methods simultaneously. Notifications are only generated when

the Audit subsystem failure action field is set to Log warning or Terminate server.

Procedure

1. Optional: Click **Security** > **Security Auditing**.
2. Optional: Confirm the Audit subsystem failure action field is set to Log warning or Terminate server. If the Audit subsystem failure action field is set to No warning, then notifications will not be generated.
3. Click **Security** > **Security Auditing** > **Audit monitor** .
4. Under Notifications, Click **New**
5. Enter the name that should be associated with this notification configuration in the Notification name field.
6. Select the Message log check box to specify the failure notifications are recorded in the audit log.
7. Select the email sent to notification list check box to specify that failure notification email should be sent to the addresses listed in the notification list.
8. Enter an email address in the email address to add field This step is not needed if email notifications are not going to be sent.
9. Enter the mail server address in the Outgoing mail (STMP) server address. This step is not needed if email notifications are not going to be sent.
10. Click **Add >>** to add the email address and associated mail server to the email notification list.
11. Repeat steps 5 through 7 for each email address you want to specify in the email notification list.
12. Click **OK**.
13. Select the Enable monitoring check box to turn on audit failure notifications.
14. Select the notification configuration to be used from the Monitor notification dropdown menu.
15. Click **OK**.

Results

After completing this task, a notification will be generated if the security auditing subsystem experiences an unrecoverable error resulting in security events no longer being audited.

What to do next

After configuring notifications, you can analyze your audit data for potential weaknesses in the current security infrastructure and to discover possible security breaches that might have occurred.

Audit notifications cannot be removed using the administrative console. To remove an audit notification you first must run the deleteAuditNotificationMonitorByRef or the deleteAuditNotificationMonitorByName command. After running one of those commands, remove the audit notification by running the deleteAuditNotification command.

Audit monitor collection

Use this page to configure audit subsystem failure notifications. The Auditor monitor panel lists the existing notification configurations and is the gateway for creating new notification configurations and for managing the existing notification configurations.

To view this administrative console page, click **Security > Security Auditing > Audit monitor**.

Enable monitoring:

Specifies whether to enable or disable notifications. If the check box is selected, then monitoring is enabled. If the check box is not selected, then monitoring is disabled. This check box is disabled by default.

Monitor notification:

Specifies the notification configuration that will be used for reporting audit subsystem failures.

Notification name:

Specifies a string that uniquely identifies a notification configuration.

Message log:

Specifies if the configuration will send failure notifications to the message log file. If the value is true, then failure notifications will be sent to the message log file. If the value is false, then failure notifications will be not be sent to the message log file. When creating a notification, this field is in the form of a check box and is not selected by default.

Send Email:

Specifies whether an email notification is sent to the addresses listed in the List of email addresses column.

List of email addresses:

Specifies the email addresses listed as recipients for email notification in the event of an audit subsystem failure. No email addresses are listed by default. Email addresses will appear in this column if they are listed in the notification list in the notification, this applies even when the Email sent to notification list check box is not selected in the notification.

Audit notification settings

Use this page to create and manage notification configurations that define how auditors are made aware of audit subsystem failures.

To view this administrative console page, click **Security > Security Auditing > Audit monitor > New**.

Notification name:

Specifies a string that uniquely identifies a notification configuration.

Message log:

Specifies if the configuration will send failure notifications to the message log file. If the check box is selected, then failure notifications will be sent to the message log file. If the check box is not selected, then failure notifications will be not be sent to the message log file. This check box is not selected by default.

Send secure emails:

Email sent to the notification list:

Specifies whether the configuration will send a failure notification to the recipients listed in the notification list. If the check box is selected, then failure notifications will be sent to the recipients in the notification list. If the check box is not selected, then failure notifications will not be sent to the recipients in the notification list. This check box is not selected by default.

Email address to add:

Specifies the email address to be added to the notification list to received failure notification emails. To add a recipient to the notification list, this field and the Outgoing mail (SMTP) server field must both be completed before you click the **Add**.

Outgoing mail (SMTP) server:

Specifies the SMTP server to be used with this email address. If no server is specified, then the email realm will be used.

Configuring the default audit service providers for security auditing

The audit service provider is used to format the audit data object that was sent by the audit event factory. After being formatted, the audit data is recorded to the repository defined in the audit service provider configuration.

Before you begin

Before configuring the audit service provider, enable global security in your environment.

About this task

This task configures the audit service provider used to record generated audit records.

Procedure

1. Click **Security** > **Security Auditing** > **Audit service provider**.
2. Click **New** and then select **Binary file based emitter**.
3. Enter the unique name that should be associated with this audit service provider in the Name field.
4. Enter the file location of the binary log file in the Audit log file location field.
5. Optional: Enter the maximum size allowed for a single binary log file in the Audit log file size field.

   This field is specified in megabytes. After the maximum audit file size is reached, a new audit file will be created or an existing audit file will be overwritten. If the maximum number of audit log files has not been set, the default maximum file value used is 10 megabytes. There is no audit archiving utility included with the product. You are responsible for the archiving of your audit data.
6. Optional: In the Maximum number of audit log files field, enter the maximum number of audit logs to be stored before the oldest is overwritten.

   The default value for this field is 100. The value of 100 is also used if the field is empty.

   Note: The maximum number of logs does not include the current binary log that is being written to. It is a reference to the maximum number of archived (timestamped) logs. The total number of binary logs that can exist for a server process is the maximum number of archived logs plus the current log.

   Also under this field, there are additional options to select the behavior when the maximum number of logs is reached. The choices are:

   oldest If you select this option, when the maximum audit logs are reached, the oldest audit log is rewritten; notification is not sent to the auditor.

   stop server
   This option does not rewrite over the oldest audit log. It stops the audit service, sends a notification to the SystemOut.log, and quiesces the application server.

   stop logging
   This option does not rewrite over the oldest audit log. It also stops the audit service, but does allow the WebSphere process to continue. Notifications are not posted in the SystemOut.log.
7. Select the filters to be used by this audit service provider. The Selectable filter list consists of a list of the configured filters that have been configured and are currently enabled.

a. Select the filters that should be audited from the Selectable filter list.
b. Click **Add >>** to add the selected filters to the Enabled filter list.

8. Click **Apply**.

Results

After completing these steps, your audit data will be sent to the specified repository in the format required by that repository.

What to do next

After creating an audit service provider, the audit service provider must be associated with an audit event factory provide the audit data objects to the audit service provider. Next you should configure an audit event factory.

Audit service provider collection

The Audit service provider panel displays a listing of all configured audit service provider implementations. Using this panel, a user can define a new audit service provider implementation, delete an existing implementation, and display or modify the fields associated with an existing implementation.

To view this administrative console page, click **Security > Security Auditing > Audit service provider**.

By default, the `audit.xml` will contain the IBM audit service provider implementation which emits audit records to a binary filed-based text file. This implementation is used for Binary file- based audit service provider configurations. For each existing audit service provider in the list on this panel, the unique name, type and event formatting class associated with the audit service provider will be displayed.

Name:

The Name field is the unique name associated with the audit service provider implementation.

Type:

The Type field specifies if the implementation is a binary file-based implementation, SMF implementation or a third party implementation.

Event formatting module class name:

The event formatting class is a class used to format the generic event data object into a format that is specific to the audit service provider implementation. For example, a third party audit service provider implementation might have an event formatting class that takes the generic event and translates it into XML data. There is no Event formatting module class for binary file-based implementations nor for SMF implementations.

Audit service provider settings

Use this page to define the implementation details of the audit service provider. There are three types of audit service providers: binary file-based, third party and SMF.

To view this administrative console page, click one of the following paths:

- **Security** > **Security auditing** > **Audit service provider** > ***audit_service_provider_name***.
- **Security** > **Security auditing** > **Audit service provider** > **New** > **Binary File-based emitter**.
- **Security** > **Security auditing** > **Audit service provider** > **New** > **Third party emitter**.

Name:

Specifies the unique name associated with the audit service provider.

Third party emitter class name:

Specifies the name of the class for this implementation. This field is only present for Third party emitter implementations.

Audit file location:

Specifies the path to the binary log file.

Audit file size:

Specifies the maximum size of a single binary log file. This value is defined in megabytes.

Maximum number of audit log files:

Specifies the maximum number of binary log files to create before the oldest is replaced.

Note: The maximum number of logs does not include the current binary log that is being written to. It is a reference to the maximum number of archived (timestamped) logs. The total number of binary logs that can exist for a server process is the maximum number of archived logs plus the current log.

Audit log wrapping:

Specifies the wrapping behavior of the binary audit log when the maximum number of binary audit log files is reached.

Note: In this release of WebSphere Application Server, there are new customizable options available when specifying the default audit log wrapping behavior. This is only applicable to the Binary Audit Log implementation.

Choose from one of the following options:

WRAP
If you select this option, when the maximum audit logs are reached, the oldest audit log is rewritten; notification is not sent to the auditor. This is the default option, and mimics the default behavior in WebSphere Application Server Version 7.0.

NOWRAP
This option does not rewrite over the oldest audit log. It stops the audit service, sends a notification to the `SystemOut.log`, and quiesces the application server.

SILENT_FAIL
This option does not rewrite over the oldest audit log. It also stops the audit service, but does allow the WebSphere process to continue. Notifications are not posted in the `SystemOut.log`.

Note: If audit notification of failures in the audit subsystem is configured, and SILENT_FAIL is selected, the auditor is not notified of the audit subsystem failure. The SILENT_FAIL option takes precedence

Note: If you use the NOWRAP or SILENT_FAIL options, when the server is stopped as a result of the logs being maxed-out, a stopserver is performed, or because the server abends in some way, you must archive the binary audit logs before you restart the server.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Event formatting module class name:

Specifies a class used to format the generic event into a format that is specific to the audit service provider implementation. For example, a third party audit service provider implementation might have an event formatting class that takes the generic event and translates it into XML data.

Selectable filters:

Specifies the available event filters. To enable a filter for an implementation, select the filter from the Selectable event filters list and then click **>**.

Enabled filters:

Specifies the event filters that are currently enabled for an implementation. To disable a filter for an implementation, select the filter from the Enabled filters list and then click **<**.

Custom properties:

Specifies any custom properties that might be used to add properties to a third party implementation. Custom properties are not available for binary file-based implementations or SMF implementations.

- Name
- Value

Example: Base Generic Emitter Interface

The Base Generic Emitter interface defines how audit events are emitted. Other interfaces can exist to extend this interface and to process specific audit events groupings, such as security events, transactional events, or some other custom grouping. Use this interface to create a custom implementation of the emitter.

Base Generic Emitter Interface

```
/**
* This is the interface for the event emitter. Event sources use this interface
* to send events to an event service.
*
*/
public interface BaseGenericEmitter {
/**
* Sends an event to the configured GenericEmitter implementation.
*
* @param event The event to be sent to the event service.
* This value cannot be null.
* @return The global instance ID of the event that was built.
* @exception GenericEmitterException If an error occurs during emitter processing.
* @exception IllegalArgumentException If the event parameter is null.
*/
public String sendEvent(GenericEvent event) throws
     GenericEventException;
/** * Sends an array of events to the configured GenericEmitter implementation.
* @param events The event array to be sent to the event service.
* This value cannot be null.
* @return The global instance IDs of the events that were built.
* @exception GenericEmitterException If an error occurs during emitter processing.
* @exception IllegalArgumentException If the events parameter is null.
*/
public String[] sendEvents(GenericEvent events[]) throws
     GenericEventException;
/**
* Causes the emitter to release all resources that are owned by this
```

```
* object and its dependents.
* Subsequent calls to this method have no effect.
*
* @throws GenericEmitterException If the emitter does release the
* held resources.
* resources.
* @throws GenericEventException If any other error occurs when releasing resources.
*/
public void close() throws
     GenericEventException;
}
```

Configuring a third party audit service providers for security auditing

The audit service provider is used to format the audit data object that was sent by the audit event factory. In addition to the default audit service provider, you may use a third party implementation as your audit service provider.

Before you begin

Before configuring the audit service provider, enable global security in your environment.

About this task

This task configures the audit service provider used to record generated audit records.

Procedure

1. Click **Security** > **Security Auditing** > **Audit service provider**.
2. Click **New** and then select **Third party emitter**.
3. Enter the unique name that should be associated with this audit service provider in the Name field.
4. Enter the Third party emitter class name.
5. Enter the Event formatting module class name. This field specifies the class used to format the generic event into a format that is specific to the audit service provider implementation. For example, your implementation might have an event formatting class that takes the generic event and translates it into XML data.
6. Select the filters to be used by this audit service provider. The Selectable filter list consists of a list of the configured filters that have been configured and are currently enabled.
   a. Select the filters that should be audited from the Selectable filter list.
   b. Click **Add >>** to add the selected filters to the Enabled filter list.
7. Optional: Enter any custom properties that you included in your third party emitter code.
8. Click **Apply**.

Results

After completing these steps, your audit data will be sent to the specified repository in the format required by that repository.

What to do next

After creating an audit service provider, the audit service provider must be associated with an audit event factory provide the audit data objects to the audit service provider. Next you should configure an audit event factory.

Example: Base Generic Emitter Interface

The Base Generic Emitter interface defines how audit events are emitted. Other interfaces can exist to extend this interface and to process specific audit events groupings, such as security events, transactional events, or some other custom grouping. Use this interface to create a custom implementation of the emitter.

Base Generic Emitter Interface

```
/**
* This is the interface for the event emitter. Event sources use this interface
* to send events to an event service.
*
*/
public interface BaseGenericEmitter {
/**
* Sends an event to the configured GenericEmitter implementation.
*
* @param event The event to be sent to the event service.
* This value cannot be null.
* @return The global instance ID of the event that was built.
* @exception GenericEmitterException If an error occurs during emitter processing.
* @exception IllegalArgumentException If the event parameter is null.
*/
public String sendEvent(GenericEvent event) throws
     GenericEventException;
/** * Sends an array of events to the configured GenericEmitter implementation.
* @param events The event array to be sent to the event service.
* This value cannot be null.
* @return The global instance IDs of the events that were built.
* @exception GenericEmitterException If an error occurs during emitter processing.
* @exception IllegalArgumentException If the events parameter is null.
*/
public String[] sendEvents(GenericEvent events[]) throws
     GenericEventException;
/**
* Causes the emitter to release all resources that are owned by this
* object and its dependents.
* Subsequent calls to this method have no effect.
*
* @throws GenericEmitterException If the emitter does release the
* held resources.
* resources.
* @throws GenericEventException If any other error occurs when releasing resources.
*/
public void close() throws
     GenericEventException;
}
```

Configuring audit event factories for security auditing

The audit event factory collects the data associated with the auditable security events and builds the audit data object. The object is then sent to the audit service provider to be formatted and recorded to a specified repository.

Before you begin

Before configuring an event factory, enable global security in your environment. An event type filter and an audit service provider need to be created before completing these steps

About this task

Procedure

1. Click **Security** > **Security Auditing** > **Audit event factory configurations** > **New**.
2. Enter the unique name that should be associated with this Audit event factory configuration in the Name field.
3. Select either **IBM audit event factory** or **Third party event factory**.
   a. Enter the Third party audit event factory class name. This step is only required if a Third party event factory is being created.
4. Select the appropriate audit service provider implementation from the Audit service provider dropdown menu,
5. Select the event type filter configuration to be used by this audit event factory. The Filters list consists of a list of the event type filter configurations that have been created and are currently enabled.
   a. Select the event type filters that should be used from the Selectable filter list.
   b. Click **Add >>** to add the selected event type filter configurations to the Enabled filter lists.
6. Enter any Custom properties that need to be included with this audit event factory configuration. Custom properties are only available for Third party event factory implementations.

7. Click **Apply**.

Results

After successful completion of these steps, you will have an event factory that can be used to gather auditable event data.

What to do next

After configuring an audit event factory, you can optionally protect your data by configuring the security auditing subsystem to sign and encrypt your audit logs.

Audit event factory configuration collection

The Audit event factory configuration panel displays a list of all currently configured audit event factory implementations. This panel allows a user with the auditor role to manage their configured audit event factories. This includes the ability to configure a new implementation, which is done using the **New** button on this panel.

To view this administrative console page, click **Security > Security Auditing > Audit event factory configuration**.

Name:

The Name field specifies the unique name associated with the audit event factory configuration.

Type:

The Type field specifies this audit event factory configuration as either an IBM audit event factory or a Third party audit event factory.

Class name:

The Class name field specifies the class that is being implemented in an audit event factory configuration.

The class name is `com.ibm.ws.security.audit.AuditEventFactoryImpl` for an IBM event factory. For a Third party audit event factory, the class name is the class specified in the Third party audit event factory class name field.

Audit event factory settings

The Audit event factory settings panel displays the details of a specific audit event factory. The auditor uses this panel to manage and create audit event factory configurations.

To view this administrative console page, click on of the following paths:

- **Security** > **Security Auditing** > **Audit event factory configuration** > ***audit_event_factory_configuration_name***.
- **Security** > **Security Auditing** > **Audit event factory configuration** > **New**.

Name:

Specifies the unique name associated with the audit event factory configuration.

Type:

Specifies this audit event factory configuration as either an IBM audit event factory or a Third party audit event factory. This field does not appear on the panel during the creation of a new audit event factory. It is included when viewing or modifying an existing audit event factory.

IBM audit event factory:

Specifies that the Type field of this audit event factory is IBM audit event factory. This check box only appears on the panel during the creation of a new audit event factory. This check box is selected by default when creating a new audit event factory.

Third party audit event factory:

Specifies that the Type field of this audit event factory is Third party audit event factory. This check box only appears on the panel during the creation of a new audit event factory. This check box is not selected by default when creating a new audit event factory.

- The Third party audit event factory class name field is active when the Third party audit event factory check box is selected. This field represents the class name of the third-party implementation of the Audit Event Factory interface

Class name:

Specifies the class that is being implemented in a audit event factory configuration.

Although not specified during creation, the class name is `com.ibm.ws.security.audit.AuditEventFactoryImp` for an IBM event factory.

Audit service provider:

Specifies where the audit data objects gathered by this audit event factory will be sent.

Selectable filters:

Specifies the filters that are currently available to be used for an implementation. To enable a filter for an implementation, select the filter from the Selectable filter list and then click **>**.

Enabled filters:

Specifies the filters that are currently enabled for an implementation. To disable a filter for an implementation, select the filter from the Enabled filter list and then click **<**.

Custom properties:

Specifies properties that the auditor can define to configure the Audit Event Factory implementation. This might be used by third party implementation of the audit event factory interface. Custom properties are not used for the IBM audit event factory implementation.

Each custom property has the following fields:

- Name
- Value

Example: Generic Event Factory Interface

This interface is used for processing generic audit events. Other interfaces can be defined which extend this interface to process specific audit event groupings, such as security events, transaction events, or some other custom grouping.

Generic Event Factory Interface

```
/**
* GenericEventFactory is the interface that is used to generate audit events.
* This interface may be extended to generate application specific audit events.
*
* One or more GenericEventFactory implementations each with a unique name can be defined in the
* security configuration and be used by WebSphere Application Server security auditing service.
```

```
* @author IBM Corporation
* @version WAS 7.0
* @since WAS 7.0
*/
public interface GenericEventFactory {
/**
* The init method allows a GenericEventFactory implementation to
* initialize its internal auditing configuration using the properties and context object.
*
* The properties and context objects are treated as read-only and must not be modified by the
* GenericEventFactory implementation.
*
* @param A String object represents the name of this GenericEventFactory.
* @param A Map properties object that contains the custom properties that can be defined in the
* the admin console or by using wsadmin scripting tool.
* @param A Map object that contains the context that includes cell name, node name, and server name.
* @exception ProviderFailureException might occur if the audit factory does not initialize
*/
public void init(String name, Map properties, Map context) throws ProviderFailureException;
/**
*
* The terminate method gracefully quiesces the event factory implementation.
*/
public void terminate();
/**
*
* The refresh method allows a GenericEventFactory implementation to
* update its internal auditing configuration using the properties object.
*
* The properties object is treated as read-only and must not be modified by the
* GenericEventFactory implementation.
*
* @param A Map object that contains the custom properties
* @exception ProviderFailureException might occur if the factory does not refresh
*/
public void refresh(java.util.Map properties) throws ProviderFailureException;
/**
*
* The getName method returns the name of this GenericEventFactory.
*
* @param None
* @return a String object represents the name of the GenericEventFactory.
*/
public String getName();
/**
*
* The sendEvent method determines whether the specified audit event is generated by this
* GenericEventFactory.
*
* @param a String object represents an audit event
* @param a OutcomeType object represents the audit outcome value
* @exception ProviderFailureException might occur if the audit factory does not initialize
* @return a boolean success/failure
* @exception ProviderFailureException might occur if the audit factory does not send the event.
*/
public boolean sendEvent(String auditEventType, OutcomeType auditOutcome) throws
ProviderFailureException;
*
}
```

Protecting your security audit data

The security auditing subsystem allows for protection of your security audit data by increasing the assurance that the audit data has not been tampered or modified outside of the auditing facility. This option also protects the confidentiality of the data. The audit data is protected by encrypting and signing the recording data.

Before you begin

Restriction: Signing and encrypting your audit data is only available for data created using the default binary log audit service provider. If you are using the SMF emitter or a 3rd party emitter you will not be able to sign or encrypt your data.

Before configuring protection for your security audit data, enable global security and security auditing in your environment. You must be assigned the auditor role to complete the task of protecting your audit data. You will also need the administrator role to configure your audit data to be signed.

About this task

The practice of auditing requires assurances that your audit data is accurate and uncompromised. Your audit data has the option to be encrypted, signed, or encrypted and signed. You can protect your audit data using these options to provide assurances that you data is only viewed by authorized users and can not untraceably be modified . To protect the validity of your security auditing functionality, complete the following steps:

Procedure

1. “Encrypting your security audit records” Audit logs can be encrypted to ensure your audit data is protected. The audit logs will be encrypted using a certificate that is saved to a keystore in the `audit.xml` file. By encrypting your audit records, only users with the password to the keystore will be able to view or update the audit logs.
2. “Signing your security audit records” on page 1874 Audit logs can be signed to ensure the integrity of your audit data. By signing your audit records, you ensure any modifications of the audit logs can be traced.

Results

After completing these steps your data will be signed, encrypted or signed and encrypted to provide assurances that the data is accurate and confidential.

What to do next

After protecting your data, you can configure notifications to ensure you are notified if a problem with the security auditing subsystems occurs that prevents security events from being recorded.

Encrypting your security audit records

Audit logs can be encrypted to ensure your audit data is protected. By encrypting your audit records, only users with access to the encrypting certificate will be able to view the audit logs.

Before you begin

Restriction: Encrypting audit data is only available for data created using the default audit service provider. If you are using the SMF emitter or a 3rd party emitter you will not be able to encrypt your data.

Before configuring your security audit records to be encrypted, enable global security and security auditing in your environment. You must be assigned the auditor role to encrypt your security auditing records. If you are using a certificate stored in the `security.xml` file, you also require the administrator role to complete this task.

About this task

Procedure

1. Click **Security** > **Security Auditing** > **Audit record encryption configuration**.
2. Select the Enable encryption check box to specify that your audit records should be encrypted. All other fields on this panel will be unavailable until this check box has been selected.
3. Select the keystore that contains the encrypting certificate from the dropdown menu or click **New** to create a new certificate in an existing keystore. Use the following steps if you are creating a new certificate:
   a. Enter the name of the keystore in the Name field.
   b. Enter the path to the keystore file in the Path field.
   c. Enter the password to be associated with the keystore in the Password field.

d. Confirm the password associated with the keystore by retyping the password in the Confirm password field.
e. Select the keystore type from the Type dropdown list. The default value of the Type dropdown list is PKCS12.

4. If you are using an existing certificate to encrypt your audit records, ensure **Certificate in keystore** is selected and specify the intended certificate in the Certificate alias dropdown menu.
5. If you are generating a new certificate to encrypt your audit records, select **Create a new certificate in the selected keystore** and follow these steps:
   a. Enter the name of your new certificate in the Certificate alias field.
   b. Select either Automatically generate certificate or Import a certificate. The certificate used to encrypt the data in the audit log files can either be created or imported. If you selected to generate a certificate, then skip to the last step on this page. If you selected to import a certificate, then continue on with step c.
   c. Enter the name of the keystore file in the Key file name field.
   d. Enter the path to the keystore file in the Path field.
   e. Select the keystore type from the Type dropdown list. The default value of the Type dropdown list is PKCS12.
   f. Enter the password associated with the keystore in the Key File password field.
   g. Click **Get key file aliases** to populate the Certificate alias to import dropdown menu.
   h. Select the certificate to be imported from the Certificate alias to import dropdown menu.
6. Click **OK**.

Results

After completing these steps, your audit logs will be encrypted to ensure only authorized users can view the content of your audit log files.

What to do next

After you have finished configuring your audit logs to be encrypted, you can ensure the data integrity of your audit logs by configuring the audit subsystem to sign your audit records.

Signing your security audit records

Audit logs can be signed to ensure the integrity of your audit data. By signing your audit records, modifications of the audit logs can be traced.

Before you begin

Restriction: Signing audit data is only available for data created using the default audit service provider. If you are using the SMF emitter or a 3rd party emitter you will not be able to sign your data.

Before configuring your security audit records to be signed, enable global security and security auditing in your environment. You must be assigned the auditor role and the administrator role to configure audit record signing.

About this task

Procedure

1. Click **Security** > **Security Auditing** > **Audit record signing configuration**.
2. Select the Enable signing check box to specify that your audit records should be signed. All other fields on this panel will be unavailable until this check box has been selected.

3. Select the keystore that contains the signing certificate from the Managed keystore containing the signing certificate dropdown menu.
4. If you are using an existing certificate to sign your audit records, ensure Certificate in keystore is selected and specify the intended certificate in the Certificate alias dropdown menu.
5. If you are generating a new certificate to sign your audit records, select Create a new certificate in the selected keystore and follow these steps:
   a. Enter the name of your new certificate in the Certificate alias field.
   b. Select on of the following options: Import the encryption certificate, Automatically generate certificate or Import a certificate. The certificate used to encrypt the data in the audit log files can either be created or imported.
      - If you selected Import the encryption certificate, then you will use the encryption certificate to also sign your audit records. Skip to the last step on this page to complete this configuration.
      - If you selected to generate a certificate, then skip to the last step on this page to complete this configuration.
      - If you selected to import a certificate from an existing keystore, then continue on with step c.
   c. Enter the name of the keystore file in the Key file name field.
   d. Enter the path to the keystore file in the Path field.
   e. Select the keystore type from the Type dropdown list. The default value of the Type dropdown list is PKCS12.
   f. Enter the password associated with the keystore in the Key File password field.
   g. Click **Get key file aliases** to populate the Certificate alias to import dropdown menu.
   h. Select the certificate to be imported from the Certificate alias to import dropdown menu.
6. Click **OK**.

Results

After you have completed these steps, your audit logs will be digitally signed to ensure the integrity of the data.

What to do next

After you have finished configuring your audit logs to be signed, you can ensure the confidentiality of your audit logs by configuring the audit subsystem to encrypt your audit records.

Audit encryption keystores and certificates collection

The Audit encryption keystores and certificates panel allows the auditor to manage the keystores and certificates used for audit encryption.

To view this administrative console page, click **Security > Security Auditing > Audit encryption keystores and certificates**.

Name:

Specifies the unique name of the keystores used for storing the encryption certificate.

Path:

Specifies the path to the listed keystore file.

The path to the keystore file can be listed using environment variables, ${PROFILE_ROOT}, or with a fully qualified path.

Audit record encryption configuration settings

Use this page to enable encryption for your audit records. Encrypting your audit records ensures only a user given access to the certificate used for encryption is allowed to view the audit records.

To view this administrative console page, click **Security > Security auditing > Audit record encryption configuration**. If **Enable encryption** is not selected, then all of the other fields on this panel will be disabled. Encryption is not enabled by default.

Enable encryption:

Specifies whether your audit records will be encrypted. This check box is not selected by default.

Audit keystore containing the encryption certificate:

Specifies the audit keystore specified to store the encryption certificate.

A new keystore can be created by clicking on the **New...** button.

Certificate in keystore:

Specifies an existing certificate will be used from the keystore specified in the Audit keystore containing the encryption certificate field. This field is selected by default. If a keystore in the `security.xml` file is used, the administrator role is required.

- Certificate alias

  When the **Certificate in keystore** field is selected, the certificate alias dropdown menu displays a list of certificate aliases contained in the keystore defined by the **Audit keystore containing the encryption certificate** field. Select the certificate from the dropdown menu to be used to encrypt your audit records.

Create a new certificate in the selected keystore:

Specifies that a new certificate will be created in the keystore defined by the **Audit keystore containing the encryption certificate** field.

- Certificate alias

  When the **Create a new certificate in the selected keystore** is selected, the **Certificate alias** field is used to define the name of the certificate to be created in the keystore defined by the **Audit keystore containing the encryption certificate** field.
- Automatically generate certificate

  When selected, the **Automatically generate certificate** field specifies that the application server will automatically generate the certificate. This field is selected by default when the **Create a new certificate in the selected keystore** field is selected.
- Import a certificate

  When selected, the **Import a certificate** field specifies that an existing self-signed certificate will be imported by the auditor into the keystore and used to encrypt your audit records. This field is not selected by default when the **Create a new certificate in the selected keystore** field is selected. The following fields need to be defined to import an existing certificate.
  - The **Key file name** field specifies the keystore filename that contains the certificate to be imported.
  - The **Path** field specifies the path to the keystore file that contains the certificate to be imported.
  - The **Type** field specifies the type of the keystore file that contains the certificate to be imported.
  - The **Key file password** field specifies the password used to access the keystore file that contains the certificate to be imported.
  - **Certificate alias to import** field specifies the alias of the certificate to be imported.

Audit record signing configuration settings

Use this page to enable signing for your audit records. Signing audit records ensures tamper-proof recording of the auditable events. Both the auditor and administrator roles are required to configure the signing of your audit data.

To view this administrative console page, click **Security** > **Security auditing** > **Audit record signing configuration**. If **Enable signing** is not selected, then all of the other fields on this panel will be disabled.

Enable signing:

Specifies whether your audit records will be encrypted. This check box is not selected by default.

Managed keystore containing the signing certificate:

Specifies the keystore used to store the signing certificate.

Certificate in keystore:

Specifies an existing certificate will be used from the keystore specified in the **Managed keystore containing the signing certificate** field. This field is selected by default.

- Certificate alias

  When the **Certificate in keystore** field is selected, the **Certificate alias** dropdown menu displays a list of certificate aliases contained in the keystore defined by the **Managed keystore containing the signing certificate** field. Select the certificate from the dropdown menu to be used to sign your audit records.

Create a new certificate in the selected keystore:

Specifies that a new certificate will be created in the keystore defined by the **Managed keystore containing the signing certificate** field.

- Certificate alias

  When the **Create a new certificate in the selected keystore** is selected, the **Certificate alias** field is used to define the name of the certificate to be created in the keystore defined by the **Audit keystore containing the encryption certificate** field.
- Import the encryption certificate

  Specifies the certificate used for encryption will be imported into the signing keystore file and used for signing.
- Automatically generate certificate

  Specifies the application server will automatically generate the certificate. This field is selected by default when the **Create a new certificate in the selected keystore** field is selected.
- Import a certificate

  Specifies an existing self-signed certificate will be imported by the auditor into the keystore and used to encrypt your audit records. This field is not selected by default when the **Create a new certificate in the selected keystore** field is selected. The following fields need to be defined to import an existing certificate.
  - The **Key file name** field specifies the keystore filename that contains the certificate to be imported.
  - The **Path** field specifies the path to the keystore file that contains the certificate to be imported.
  - The **Type** field specifies the type of the keystore file that contains the certificate to be imported.
  - The **Key file password** field specifies the password used to access the keystore file that contains the certificate to be imported.
  - **Certificate alias to import** field specifies the alias of the certificate to be imported.

Audit record keystore settings

The Audit record keystore panel is used by an auditor to define the keystores used for storing the encryption certificate used to encrypt the audit records. Keystores used for auditing are managed outside of other keystores being used on the system to facilitate separation of the authority of the auditor for the authority of the administrator.

To view this administrative console page, click one of the following paths:

- **Security > Security Auditing > Audit encryption keystores and certificates >** *keystore_name*.
- **Security > Security Auditing > Audit encryption keystores and certificates > New**.
- **Security > Security Auditing > Audit record encryption configuration > New**

Name:

The Name field specifies the unique name for the keystore. This is a required field.

Path:

Specifies the path where the keystore file is located. This is a required field.

Password:

Specifies the password to be used for this keystore. This is a required field.

Confirm Password:

Specifies confirmation of the value provided in the Password field. This is a required field.

Type:

The Type field specifies the type of the keystore. The Type dropdown menu has the following options for defining the keystore type:

- JCEKS
- CMSKS
- PKCS12 - The default value for the Type field is PKCS12.
- Cryptographic Token Device (PKCS11)
- JKS
- PKCS12JarSigner

Using the audit reader

The audit reader is a utility that can be used to read the binary audit logs generated by the default binary emitter implementation. The audit reader parses the audit log to generate an HTML report. The audit reader is invoked using wsadmin commands and is not accessible using the administrative console.

Before you begin

The audit reader can only be used to parse log files that are created by the default audit service provider. Logs created by a third-party emitter can not be parsed by the audit reader.

About this task

Your audit logs might be encrypted, signed, encrypted and signed or neither encrypted nor signed. The audit reader is able to parse any of these combinations to generate an HTML report. If the audit log file is

encrypted, the password of the keystore storing the certificate used to encrypt the log must be provided. The showAuditLogEncryptionInfo wsadmin command can be used to get information to determine which keystore was used to sign the audit log.

Depending on the selections you made in your audit service provider configuration, the size of the audit logs can become large enough to make them cumbersome to review. What data has been recorded into your log is dependant on the event type filers you are using and whether you specified to use verbose logging. Options are provided for you to further limit the data included in the HTML report that is generated by the audit reader to a subset that you specify. The audit reader can be used to parse the same data multiple times to generate separate reports for your different requirements.

By default, all event types, outcome types, timestamps, and sequence numbers will be gathered from the Binary Audit log and generated into a report. The ability to specify only specific event types, only specific sequence numbers, only records with specific timestamps, as well as specific outcome types is provided. A sequence number is a unique identifier assigned to each audit record. Options exist to limit which events, outcomes, and sequence numbers are included in the report.

The report type controls what data is reported for each audit record in the log file. The default report type includes the follow data for each audit record:

- creationTime
- action
- progName
- registryType
- domain
- realm
- remoteAddr
- remotePort
- remoteHost
- resourceName
- resourceType
- resourceUniqueId

The complete report type generates a report based on all the data that was logged for the selected audit records. The complete report type includes all the data that is included by the default report type and all the additional datapoints that were logged for these audit records. The additional available datapoints for an audit record varies depending on the event type it represents.

A custom report type is also included. Use the custom report type to specify only the datapoints that you want generated in the report. A report may be generated based on the following criteria:

- all or specific event types
- all or specific outcome types
- all or a specific sequence number range
- all or a specific timestamp range

Procedure

Run the binaryAuditLogReader wsadmin command to use the audit reader to generate a log report. See the AuditReaderCommands command group for the AdminTask object article for more information.

Results

After you complete these steps, you will generated an HTML report containing the data specific to your requirement.

Example

Audit Event Outcome Codes

In a binary audit log or the output of the audit reader tool, audit event outcomes are expressed with a numeric code. Use this table to associate the audit event outcome code in the binary audit logs to a generic error messages.

Table 203. Event Outcome Codes. This table lists the event outcome codes.

Outcome Reason Code	Description
0	An error occurred while parsing the certificate.
1	The security context does not exist for the thread.
2	There is conflicting session evidence.
3	The session has been rejected.
4	The token has expired.
5	Successful authentication has occurred.
6	Successful authentication for accessing a resource has occurred.
7	Successful authentication occurred while mapping a user.
8	Successful authorization has occurred.
9	Login termination was successful.
10	Invalid evidence exists.
11	There was a GSS formatting error.
12	Credentials were unauthenticated.
13	Authentication failed.
14	An invalid resource was accessed.
15	Authentication was denied.
16	Authorization was denied.
17	Access was denied because of an authentication failure.
18	Authorization was excluded.
19	Authorization was excluded because of access without proper security role.
20	An unsupported authentication mechanism was used.
21	An authentication redirect occurred.
22	The context does not exist.
23	A TAI challenge occurred.
24	A TAI validation was not successful.
25	A TAI mapping was not successful.
26	A provider failure occurred.
27	A SSO token validation was not successful.
28	An invalid user id or password was provided.
29	A send login form
30	An invalid configuration exists.
31	An user id or password is missing.
32	Failure occurred for an unknown reason.
33	The account was disabled because of retry violations.
34	The account was locked out because of retry violations.
35	The account was locked out because the maximum number of unsuccessful login attempts has occurred.
36	The account is disabled.
37	The account has expired.

Table 203. Event Outcome Codes (continued). This table lists the event outcome codes.

38	The account is unlocked.
39	The maximum inactive time permitted for the account has elapsed.
40	The password has expired.
41	The minimum interval for a password change has unexpired.
42	The maximum interval permitted before a password must be changes has elapsed.
43	An authentication failure has occurred.
44	An invalid user name was provided.
45	A pin is required.
46	This outcome code is not used in this release.
47	A user mapping did not occur successfully.
48	A certificate failure occurred.
49	A policy violation has occurred.
50	A policy violation has occurred because of the time of day.
51	The policy allows access.
52	A policy violation has occurred because the maximum number of unsuccessful login attempts has been reached.
53	A user name mismatch has occurred.
54	An invalid user password was provided.
55	A token signature violation has occurred.
56	The token is not yet valid.
57	The token is not supported.
58	The token is not in a valid format.
59	A credential mapping failure occurred.
60	The delegate is not authorized.
61	Access to a resource is unauthorized because of an authorization.
62	Access to a resource is unauthorized because of a time of day policy.
63	Access to a resource is unauthorized.
64	Access to a resource is unauthorized because of quality of protection.
65	Access to a resource is unauthorized because of an authorization level.
66	Access to a resource is unauthorized because reauthentication is required.
67	A password error has occurred because it does not meet password standards: minimum alphabetic characters required.
68	A password error has occurred because it does not meet password standards: minimum alphanumeric characters required.
69	A password error has occurred because it does not meet password standards: minimum numeric characters required.
70	A password error has occurred because it does not meet password standards: minimum alphabetic low case characters required.
71	A password error has occurred because it does not meet password standards: minimum alphabetic upper case characters required.
72	A password error has occurred because it does not meet password standards: minimum special characters required.
73	A password error has occurred because it does not meet password standards: maximum repeated characters exceeded.
74	A password error has occurred because it does not meet password standards: contains user name
75	A password error has occurred because it does not meet password standards: reused password.
76	A password error has occurred because it does not meet password standards: contains previous password.
77	A password error has occurred because it does not meet password standards: violations in number of characters.
78	A password error has occurred because it does not meet password standards: first or last characters are numeric.
79	An illegal form login configuration exists.
80	Access is denied because of a incorrect URI.

Table 203. Event Outcome Codes (continued). This table lists the event outcome codes.

81	Start was successful
82	Stop was successful.
83	The audit subsystem has been stopped.
84	The audit subsystem has successfully been enabled.
85	The audit subsystem has had a successful policy change.
86	Delegation was successful.
87	Delegation was not successful.
88	The audit subsystem has successfully been disabled.
89	An audit subsystem has occurred because a security header is missing.
90	An audit timestamp has been confirmed.
91	A bad audit timestamp has occurred.
92	Audit confidentially has been confirmed
93	Audit confidentially cannot be confirmed.
94	An audit decryption error has occurred.
103	A login attempt has been made by a user who has already logged in successfully.

Chapter 24. Administering Service integration

This page provides a starting point for finding information about service integration.

Service integration provides asynchronous messaging services. In asynchronous messaging, producing applications do not send messages directly to consuming applications. Instead, they send messages to destinations. Consuming applications receive messages from these destinations. A producing application can send a message and then continue processing without waiting until a consuming application receives the message. If necessary, the destination stores the message until the consuming application is ready to receive it.

Enabling or disabling service integration notification events

You can monitor your service integration environment by using notification events.

About this task

Service integration notification events allow you to monitor the activity of your service integration configuration by using your own system management application. For more details see Service integration notification events.

You can enable or disable service integration notification events at either the bus or the messaging engine level. The following table illustrates the effect of the two setting levels on the enablement of notifications:

Table 204. Notification settings. The first and second columns of the table list the combinations of the bus-level and messaging engine level notification settings. The third column indicates if the notification is enabled or not based on the combination of the two notification settings.

Bus-level notification setting	ME-level notification setting	Are notifications enabled
Not set	Not set	Disabled
Not set	Enabled	Enabled
Not set	Disabled	Disabled
Disabled	Not set	Disabled
Disabled	Enabled	Disabled
Disabled	Disabled	Disabled
Enabled	Not set	Enabled
Enabled	Enabled	Enabled
Enabled	Disabled	Disabled

Procedure

1. By using the administrative console, open the **Custom properties** page for either the bus or the messaging engine for which you want to set notification events:
   - Click **Service integration -> Buses -> *bus_name* -> [Additional Properties] Custom properties** to display the custom properties for a bus.
   - Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Custom properties** to display the custom properties for a messaging engine.
2. Click **New** to create a new custom property.
3. Enter sib.event.notification as the name of the custom property. Set the value of the custom property to either “enabled” or “disabled” as required.
4. Optional: Enter a description for the custom property.
5. Click **OK** and save your changes to the master configuration.

Administering service integration buses

Application servers or clusters of application servers in a WebSphere Application Server cell can cooperate to provide asynchronous messaging services. Service integration provides asynchronous messaging services, and a group of servers or clusters that cooperate in this way is called a service integration bus. Each of the cooperating servers or clusters is made a member of the bus. In the simplest case, a service integration bus consists of a single bus member, which is one application server.

Procedure

- Service integration technologies
- Bus configurations
- “Configuring buses”
- “Operating buses” on page 1941

Configuring buses

You can configure service integration buses in a variety of ways; for example you can create and apply security to a bus and you can then add servers or server clusters to that bus.

Creating a bus

When you create a service integration bus, you add a new bus in the administrative console, and then add one or more servers or server clusters as bus members. Thereafter, you administer the bus, and its constituent bus members, as a single unit.

Before you begin

- Plan your bus topology. For more information, see Bus configurations.
- Plan your security strategy. For more information, see Service integration security planning.

About this task

You can create a bus by using the administrative console. Wizards are provided to help you add a secured bus, and add a member to the bus. A messaging engine is created for most types of bus member, and you are prompted to specify a data store. The exception is WebSphere MQ server where, because a messaging engine is not created, a message store is not required.

Procedure

1. Add a secured bus or an unsecured bus. For the steps, see “Adding buses” on page 1885.
2. Add a member to the bus. See “Adding a server as a new bus member” on page 1889. This creates a messaging engine with default properties.
3. Optional: Configure the messaging engine, if required. See “Configuring messaging engine properties” on page 1897.
4. Optional: Configure the message store for the messaging engine, if required. For a messaging engine in an application server, you can change the data store configuration. You might do this, for example, if you want to use DB2 rather than Apache Derby as the database system. For more information about changing the data store configuration, see “Configuring a JDBC data source for a messaging engine” on page 1960. You can also change the file store configuration. Refer to “Modifying file store configuration” on page 1954.
5. Restart the server. The messaging engine starts when the server starts.

Results

A new bus is created.

What to do next

You can now change the configuration of the bus; for example, by adding additional members, by creating bus destinations, and by creating links to WebSphere MQ networks. You can create other buses and, if required, connect them together.

Adding buses

You can add a new service integration bus by using the administrative console. If messaging security is enabled, security settings are configured for the bus by default. You can add a unsecured bus if you disable messaging security.

Adding a secured bus:

In this task you add a new service integration bus that is secured by default. The security settings for the bus are stored in a security domain. When you add a new bus, you can assign it to the default global security domain, the cell-level domain, or specify a custom domain that contains a set of settings that are unique to the bus, or shared with another resource.

Before you begin

- Plan the security requirements for the bus. For more information about security planning, see Service integration security planning. For more information about security domains, see Messaging security and multiple security domains.
- Stop all servers that have the SIB Service enabled. This ensures that the bus security configuration is applied consistently when the servers are restarted. For more information, see Stopping an application server.

About this task

This task uses an administrative console security wizard to add a new bus. If the wizard detects that administrative security is disabled, it prompts you to configure a user repository, and enable administrative security.

By default, connecting clients are required to use SSL protected transports to ensure data confidentiality and integrity. If you do not want clients to use SSL protected transports, you can specify that you do not require this option.

The type of security domain you can specify for the bus depends on the versions of the bus members you intend to add to the bus:

- You must specify the global domain if you want to add one or more WebSphere Application Server Version 6 bus members.
- You can specify the global, cell-level, or custom domain if you want to add WebSphere Application Server Version 7.0 or later bus members only.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of buses is displayed.
2. Click **New**.
3. Type a name for the new bus. You must choose bus names that are compatible with the WebSphere MQ queue manager naming restrictions. You cannot change a bus name after the bus is created, which means that you can only interoperate with WebSphere MQ in the future if you use compatible names. See the topic about WebSphere MQ naming restrictions in the related links.
4. Ensure that the **Bus security** check box is selected.
5. Click **Next**. The Bus Security Configuration wizard is started.
6. Read the Introduction panel, and click **Next**.

7. If the wizard detects that administrative security is disabled, follow the prompts to select, and configure the appropriate user repository.
8. Click **Next**. A summary of the administrative security settings for the bus is displayed.
9. Review the summary, and click **Finish**. Administrative security for the cell is now enabled.
10. If you do not want clients to use SSL protected transports, clear the check box **Require clients use SSL protected transports** .
11. Select a security domain for the bus.
12. If you have selected to use a custom security domain, follow the prompts to specify a user realm.
13. Review the summary of your choices, and click **Finish**.
14. Save your changes to the master configuration.

Results

You have created a new bus secured with your chosen security settings.

What to do next

- You must restart the servers. Starting an application server provides more information.
- You can add bus members to the bus.
- Groups of users in the user repository require explicit authority to access the bus. For more information, see Administering authorization permissions.

Adding an unsecured bus:

By default, bus security is enabled and you add buses with security configured. However, you can add a new service integration bus without any security settings. If you want to secure the bus at a later date, you can do so by configuring the security settings for the bus.

About this task

To add a bus, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of buses is displayed.
2. In the content pane, click **New**.
3. Type a name for the new bus. You must choose bus names that are compatible with the WebSphere MQ queue manager naming restrictions. You cannot change a bus name after the bus is created, which means that you can only interoperate with WebSphere MQ in the future if you use compatible names. See the topic about WebSphere MQ naming restrictions in the related links.
4. Clear the **Bus security** check box to disable bus security.
5. Click **Next**.
6. Review the summary of the settings for the new bus, then click **Finish**.
7. Save your changes to the master configuration.

Results

Bus security is disabled, and you have added a new unsecured service integration bus.

What to do next

You can now add servers to the bus.

Configuring bus properties

You can configure how many messages the bus can handle, when to discard messages, and which messaging engines the bus can communicate with. You can also specify changes that can be made to the bus that do not require a restart of the messaging engines.

About this task

You configure bus properties after you have created a bus. To configure the properties of a bus, use the administrative console to complete the following steps:

Procedure

1. Click **Service integration -> Buses -> *bus_name***.
2. Specify the following properties for the bus:

   Description
   A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

   Inter-engine transport chain
   The transport chain used for communication between messaging engines in this bus.

   The transport chain must correspond to one of the transport chains defined in the *Messaging engine inbound transports* settings for the server. All servers automatically have a number of transport chains defined to them, and it is also possible to create new transport chains.

   When you specify the name of a transport chain, that chain must be defined to all servers hosting messaging engines in the bus. Otherwise, some messaging engines might not be able to communicate with their peers in the bus.

   When the use of permitted chains is enforced and a protocol is not specified for intra-bus communications then InboundSecureMessaging is assumed instead of InboundBasicMessaging. This can be overridden by setting the protocol attribute in the bus configuration. If InboundSecureMessaging is not a permitted chain then an error occurs.

   Discard messages
   Whether messages on a deleted message point should be retained at a system exception destination or can be discarded. Select this option to indicate that after a queue has been deleted, any messages left in the data store for that queue should be discarded.

   Configuration reload enabled
   Select this option to enable automatic update of configuration information on all the messaging engines on the bus.

   Changes to bus destinations or mediations are applied when destinations or mediations are added to or removed from the bus.

   Changes to the modifiable configuration information for any foreign bus connections are also updated automatically. The time when these changes take effect varies:

   Foreign Bus Connection properties
   Immediately

   WebSphere MQ link properties
   On channel restart, except Description (immediately), and Initial State (on messaging engine restart)

   MQ sender channel properties
   On channel restart, except Initial State (on messaging engine restart or sender channel creation)

MQ receiver channel properties
On channel restart, except Initial State (on messaging engine restart or receiver channel creation)

Publish/subscribe broker profile (0 to n) properties
Immediately

Service integration bus link properties
On link restart, except Description (immediately), and Initial State (on messaging engine restart or link creation)

To ensure that dynamic configuration updates are made on an application server, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] SIB service** then select **Configuration reload enabled**.

To ensure that dynamic configuration updates are made to each node, click **System administration -> Console Preferences** to display the Console Preferences window then select **Synchronize changes with nodes**.

Default messaging engine high message threshold
A threshold above which the messaging system will take action to limit the addition of more messages to a message point. When a messaging engine is created on the bus, the value of this property is used to set the default high message threshold for the messaging engine.

3. Specify topology, destination resources, web services and additional properties for the bus, as required. To restrict the number of audit messages, add a custom property called `audit.bus.authentication` with one of the following values:
   - `all` (audit all attempts to authenticate to the bus)
   - `failure` (audit only failure to authenticate to the bus)
   - `none` (do not audit any attempts to authenticate to the bus)
4. Click **OK**.
5. Save your changes to the master configuration.

Listing the buses

You can view the list of the service integration buses that currently exist. You can decide which buses you want to change, for example, to add a server to a bus.

About this task

To list the buses, use the administrative console to complete the following step. After you list the service integration buses, you can add or delete a bus. Also, you can select a bus to view its details, add a member to the bus, create a foreign bus connection, or apply security to the bus.

Procedure

In the navigation pane, click **Service integration -> Buses**.

Results

A list of buses is displayed in the content pane.

What to do next

You can now add or delete buses, or click on a bus name to display or configure its properties.

Displaying the topology of a service integration bus

You can display a tree view of the members of a bus, and the messaging engines used by a selected bus, in the administrative console. You can also view the runtime status for each messaging engine.

About this task

Use this task to display a *local topology* view of the bus members and messaging engines in a service integration bus. You can use this view to add servers as members of the bus.

To display the local topology view of a service integration bus, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of buses is displayed in the content pane.
2. In the content pane, click the name of the bus. For example, `SCA.SYSTEM.localhostCell01.Bus`
3. Click the tab **Local Topology**

Results

The topology of the bus is displayed as an expandable tree.

What to do next

You can expand nodes of the tree to display the bus members and their messaging engines.

You can add servers as members of the bus.

Deleting a bus

You can delete a service integration bus, for example if it is no longer in use.

Before you begin

Attention: When you delete a bus, each messaging engine on the bus is also deleted and any associated messages are discarded. If you subsequently create a new bus with the same name as the deleted bus, the new bus will not have the same universal unique identifier (UUID) as the deleted bus.

About this task

To delete a bus, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of buses is displayed in the content pane.
2. In the content pane, select the bus that you want to delete.
3. Click **Delete**.
4. Click **OK**.
5. Save your changes to the master configuration.

What to do next

You must disable the SIB Service at server startup.

Configuring the members of a bus

Use the administrative console to add, remove, and list the members of a bus.

Adding a server as a new bus member:

The members of a service integration bus are the application servers and clusters within which messaging engines for that bus can run. When you add a new bus member, you configure its message store, which is either a file store or a data store.

About this task

If you add a server as a member of a bus, WebSphere Application Server creates a messaging engine for the server. By default, the messaging engine is configured to use a file store. If you choose a data store, you have the choice of using the default JDBC data source and Derby JDBC Provider for its data store. If you do not want to use the default data source configuration, you can choose to use a different data source or you can configure the data store to use a different JDBC provider.

If you subsequently delete a bus member and then recreate it, you should make sure that you understand the life cycle of the file store or a data store. Refer to Data store life cycle and "Removing a messaging engine from a bus" on page 1898 for details.

If you are working in a mixed-version cell, a service integration bus running in this version of the product can only include WebSphere Application Server Version 6 bus members that are running in the following versions of the product:

- 6.0.2 (Fix Pack 23 or later)
- 6.1.0 (Fix Pack 13 or later)

If security is enabled, and the bus has mixed-version bus members, the bus members establish trust by using an inter-engine authentication alias. If you add a server as a bus member at WebSphere Application Server Version 6, and it is the first bus member at this level, you must select or create an authentication alias during this task. This action sets the inter-engine authentication alias.

You can optionally tune the initial and maximum Java virtual machine (JVM) heap sizes. Tuning the heap sizes helps to ensure that application servers hosting one or more messaging engines are provided with an appropriate amount of memory for the message throughput you require.

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Bus members**. A list of members in the bus is displayed.
2. Click **Add**.
3. Select scope of the new bus member: this is one of *Server*, *Cluster* or *WebSphere MQ server*. **Server** is selected by default. Only select the **Cluster** scope in WebSphere Application Server environments that support server clusters.
4. Make your selection and click **Next**.
5. Select the type of message store: it is either a file store or a data store. For more information, see File stores and Data stores. **File store** is selected by default.
6. Click **Next**.

   Optional: If you use a file store and want to change the default values, you can change them here. For more information refer to "Modifying file store configuration" on page 1954.

   Optional: If you use a data store and want the messaging engine in the bus member to use a non-default data source, select **Use existing data source** and enter the JNDI name of an existing data source, and the name of the schema and authentication alias to be used. For more information, see "Configuring a messaging engine to use a data store" on page 1957. If there are multiple messaging engines, you must configure each messaging engine to use a unique schema, otherwise FFDC error messages stating that `Connection cannot be provided as Datasource has been disabled!` might appear. This applies to DB2 in particular.

7. Click **Next**.

   Optional: You can view the current settings of the initial and maximum Java Virtual Machine (JVM) heap sizes. If you want to tune performance by changing the current settings, select the **Change heap sizes** check box and enter the required changes in the **Proposed heap sizes** fields.
8. Click **Next**.
9. If security is enabled, and adding this server creates a mixed-version bus, the wizard prompts for an authentication alias. Do one of the following:
   - Select an existing authentication alias.
   - Create a new authentication alias. Specify a unique alias name and password.

   This action sets the inter-engine authentication alias.
10. Click **Finish** to confirm the creation of the bus member.
11. Save your changes to the master configuration. You must restart the server for the changes to take effect.

Results

The member is added to the bus and a messaging engine is created for that member.

What to do next

Next, you can configure the messaging engine. For more information about configuring messaging engines and their message stores, see the related tasks.

Adding a WebSphere MQ server as a member of a bus:

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. A WebSphere MQ server bus member is used as a bus member for assigning queue points and mediation points to WebSphere MQ queues.

Before you begin

Get details of the client connection from your WebSphere MQ administrator.

Ensure that the WebSphere MQ server has been configured, that the bus has been defined and that the server is not already a member of the bus.

Decide which method to use to configure these resources. You can add the WebSphere MQ server as a bus member by using the administrative console as described in this task, or by using the “addSIBusMember command” on page 2268.

About this task

When you add a WebSphere MQ server to one or more buses, messaging engines on these buses can access queues on the target WebSphere MQ installation. When you make the server a bus member, you can override the server connection settings with settings that are specific to the new bus member. This can be useful in a multiple bus topology.

Procedure

1. Start the administrative console.
2. Navigate to the list of bus members for the bus to which you are adding the WebSphere MQ server. Click **Service integration -> Buses -> *bus_name* -> [Topology] Bus members**.

3. Click **Add**. The “Add a new bus member” wizard is displayed.
4. Select the WebSphere MQ server to add to the bus:
   a. Select **WebSphere MQ server**.
   b. From the drop-down list, select the server to add.
   c. Click **Next**.
5. Specify the virtual queue manager name.

   When sending messages to WebSphere MQ, the WebSphere MQ gateway queue manager sees the bus as a remote queue manager. The virtual queue manager name is the name that is passed to WebSphere MQ as the name of this remote queue manager. The default value is the name of the bus. If this value is not a valid name for a WebSphere MQ queue manager, or if another WebSphere MQ queue manager already exists that has the same name, then replace the default value with another value that is a valid and unique name for a WebSphere MQ queue manager. To be valid, the name must meet the following criteria:
   - It must contain between 1 and 48 characters.
   - It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).
6. Optional: To override the server connection settings, select the **Override WebSphere MQ server connection properties** check box.

   When you select this option, the connection properties for the server are made available so that you can change them to settings that are specific to this bus member. For more information about these connection properties, see “WebSphere MQ server bus member [Settings]” on page 2250.
7. Optional: If you have changed the server connection settings, you can click **Test connection** to test the connection to the associated WebSphere MQ network.
8. Click **Next**.
9. Click **Finish** to confirm.
10. Save your changes to the master configuration.

What to do next

You are now ready to create a WebSphere MQ queue-type destination for the new bus member.

Listing the members of a bus:

You can display a list of servers that have been added as members of a bus.

About this task

You can list the members of a bus, then you can add, remove, or edit the bus members. To list the members of a bus, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of buses is displayed in the content pane.
2. In the content pane, select the bus whose members you want to list.
3. In the content pane, under **Topology**, click **Bus members**. A list of members of the bus is displayed.

Removing a member from a bus:

You can remove an application server from a bus so that it is no longer used to process messages.

Before you begin

Attention:

- When you remove a member from a bus, all messaging engines on that bus member are also deleted and their associated messages are discarded. If you add the same bus member again, first you must manually delete the old data source for the messaging engines to ensure that once the new messaging engines are created, they can restart.
- If you remove a member from a bus, and it is the only member for that server, you must also disable the SIB Service at server start up.

About this task

To remove a member from a bus, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of buses is displayed in the content pane.
2. In the content pane, select the bus from which you want to remove the member.
3. In the content pane, under **Topology**, click **Bus members**. A list of members in the bus is displayed.
4. Select the bus member that you want to remove.
5. Click **Remove**.
6. Save your changes to the master configuration.

Results

The member is removed from the bus. All the messaging engines on that bus member and the core group policies that are associated with those messaging engines are deleted.

What to do next

If you have removed the only bus member for a server, you must now disable the SIB Service at server start up.

Disabling the service integration service:

The SIB Service, which provides messaging capability, is enabled automatically when you add a server to a service integration bus. If required, you can disable service integration bus functions when the application server starts.

About this task

You can choose to disable the SIB Service, for example if you have removed the only bus member for a server.

You can use the administrative console to disable the SIB Service, or wsadmin scripting. The following example shows a Jython script for disabling the SIB Service:

```
server = AdminConfig.getid('/Server:server1/')
sibService = AdminConfig.list('SIBService', server)
AdminConfig.modify(sibService, [["enable", "false"]])
```

To use the administrative console to disable the SIB Service, complete the following steps.

Procedure

1. Click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] SIB service** to display the SIB service settings pane.
2. Click the Configuration tab to display configuration properties for the service integration service.
3. Clear the Enable service at server startup check box.
4. Save your changes to the master configuration.

Results

You have disabled the SIB Service. Service integration bus functions are not available when the application server starts up.

Administering bootstrap members for a bus

A service integration bus can have bootstrap members. These are bus members, cell members for which the Service Integration Bus Service is enabled, or nominated cell members that can service requests to bootstrap into the bus, depending on the bootstrap policy configured for the bus.

Before you begin

Note: If the bus is in a mixed-version cell, the deployment manager must be at a Version 7.0 or later level.

About this task

You can configure the bootstrap member policy for a bus, nominate, list and delete bootstrap members for a selected bus as follows:

Configuring a bootstrap member policy for a bus:

By configuring a bootstrap member policy for a selected service integration bus, you control which servers respond to bootstrap requests from client applications.

Before you begin

Note: If the bus is in a mixed-version cell, the deployment manager must be at a Version 7.0 or later level.

About this task

A bus can use bootstrap members to service connection requests from client applications. In this task, you define a bootstrap member policy for a selected bus that defines which servers can bootstrap into the bus. The default bootstrap member policy is all cell members that have the Service Integration Bus Service enabled. You can restrict bootstrap membership to bus members only, or to bus members, and nominated bootstrap members. These are servers that are not bus members, but have been nominated as bootstrap members.

Procedure

1. Log onto the administrative console.
2. Click **Service integration -> Buses -> *bus_name***. The configuration panel for the selected bus is displayed.
3. Choose one of the following bootstrap members policies for the bus:

   All members of the cell with the Service Integration Bus Service enabled
   This is the default option. All the servers in the cell that have the Service Integration Bus Service enabled are bootstrap members.

Bus members and nominated bootstrap members
Bootstrap membership is restricted to servers that are bus members, or have been nominated as bootstrap members.

Bus members only
Bootstrap membership is restricted to servers that are bus members.

4. Click **Apply**.
5. Save your changes to the master configuration.

Results

Requests from client applications to connect to the bus are serviced by the servers in the cell that meet the criteria set by the bootstrap member policy you have configured for the bus.

What to do next

Use the administrative console to list, nominate or delete bootstrap members for the selected bus.

Listing the bootstrap members for a bus:

Use this task to find out which servers in the cell are available to service requests from clients to connect to a particular service integration bus. The list of available servers is restricted by the bootstrap member policy configured for the bus.

Before you begin

You must ensure that the appropriate bootstrap member policy has been configured for the bus. For more information, see “Configuring a bootstrap member policy for a bus” on page 1894.

About this task

In this task, you use the administrative console to list the bootstrap members for a selected bus. The list of bootstrap members depends on which of the following bootstrap member policies is configured for the bus:

All members of the cell with the Service Integration Bus Service enabled
This is the default policy. The list includes all the WebSphere Application ServerVersion 7.0 or later servers in the cell that have the Service Integration Bus Service enabled.

Bus members and nominated bootstrap members
The list includes all the servers that are bus members, or have been nominated as bootstrap members.

Bus members only
The list includes only servers that are members of the bus.

Procedure

1. Log into the administrative console.
2. Click **Service integration -> Buses -> *bus_name* -> [Topology] Bootstrap members**.

Results

A list of the bootstrap members for the selected bus is displayed.

What to do next

You can use the administrative console to nominate or remove bootstrap members, or configure a different bootstrap member policy for the bus.

Nominating bootstrap members for a bus:

You can nominate one or more servers on cell nodes as bootstrap members for a selected bus. When the Bus members and nominated bootstrap members policy is configured for the bus, the nominated bootstrap members can service requests from client applications to connect to the bus.

Before you begin

Note: You must ensure that the following requirements are met:

- If the bus is in a mixed-version cell, the deployment manager must be at a Version 7.0 or later level.
- The servers that you want to nominate as bootstrap members must be on Version 7.0 or later cell nodes.

About this task

In this task, you use an administrative console wizard to configure the **Bus members and nominated bootstrap members** policy, and nominate bootstrap members for a selected bus.

Procedure

1. Log onto the administrative console.
2. Click **Service integration -> Buses -> *bus_name* -> [Topology] Bootstrap members**. The Bootstrap members panel displays all the bus members, servers in the cell that have the Service Integration Bus Service enabled.
3. Select the policy **Bus members and nominated bootstrap members**.
4. Click **Apply**. A list of current bus members, and nominated bootstrap members is displayed.
5. Click **New**. The Nominate a bootstrap server wizard is started.
6. Type the name of the server that you want to add in the **Name** field. The first match in the list of available servers and clusters is highlighted. Click **Add** to add the named server to the list of servers to add.
7. When you have listed all the servers that you want to add, click **Next**.
8. Review the summary of the servers that you have chosen to add.
9. If you want to change your selections:
   a. Click **Previous** to return to the first step of the wizard.
   b. Make the changes you require, and click **Next**.
10. Click **Finish** to add the new bootstrap members to the local configuration.
11. Save your changes to the master configuration. An updated list of bus members and nominated bootstrap members is displayed.

Results

The nominated servers can service requests from client applications to bootstrap into the bus.

Deleting nominated bootstrap members from a bus:

Deleting a nominated bootstrap member from a selected service integration bus prevents the server from servicing requests from client applications to bootstrap into the bus.

Before you begin

You must configure the **Bus members and nominated bootstrap members** policy for the selected bus. For more information, refer to “Configuring a bootstrap member policy for a bus” on page 1894.

About this task

In this task, you use the administrative console to list all the bootstrap members for a selected bus, and then delete the nominated bootstrap members that you no longer want to use to bootstrap into the bus. Note that you can delete only nominated bootstrap members.

Procedure

1. Log onto the administrative console.
2. Click **Service integration -> Buses -> *bus_name* -> [Topology] Bootstrap members**. All the nominated bootstrap members for the bus are listed.
3. Select the nominated bootstrap members that you no longer want to use to bootstrap into the bus, and click **Delete**. The Bootstrap members panel displays an updated list of bootstrap members.
4. Save your changes to the master configuration.

Results

You cannot use the deleted nominated bootstrap members to bootstrap into the selected bus.

Configuring messaging engines

You can configure messaging engines in a variety of ways. For example, you can create and apply security to a messaging engine, then use this engine to send and receive messages. When you add a server cluster to a service integration bus, at least one messaging engine is created automatically. If you also use messaging engine policy assistance, some configuration properties are set automatically.

Configuring messaging engine properties:

You can configure the properties of a messaging engine in the administrative console. For example you can select whether the messaging engine is started automatically when its associated application server is started, how many messages it can process, and target groups that the engine can join.

About this task

In most cases, you can configure the properties of a messaging engine without interrupting the processing of messages by the messaging engine.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name***.
3. Configure the messaging engine properties. For information about the properties that you can configure, see the property descriptions in “Messaging engines [Settings]” on page 2124
4. Click **OK**.
5. Save your changes to the master configuration.

Listing the messaging engines in a bus:

You can view the list of existing messaging engines in a bus by using the administrative console. You can decide which messaging engines you want to change, for example which buses they are associated with.

About this task

To list the messaging engines in a bus, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that your messaging engine belongs to.
3. In the content pane, under **Topology**, click **Messaging engines**. The list of messaging engines in the bus is displayed.

Removing a messaging engine from a bus:

You can remove a messaging engine from a service integration bus if you no longer require it to send and receive messages on the bus.

Before you begin

You should be wary of deleting and recreating messaging engines on bus members for which WS-Notification-administered subscribers have been configured, because in some cases this can leave the remote web service subscription active (and passing notification messages to the local server) even though there is no longer any record of it. For more information, see the WS-Notification troubleshooting tip Problems can occur when deleting administered subscribers and messaging engines.

Procedure

1. Stop the messaging engine. You can stop either in **Immediate** or **Force** mode, as described in "Stopping a messaging engine" on page 1951.
2. Use the wsadmin command deleteSIBEngine to delete the messaging engine. All service integration bus links, MQ links, and custom properties that are owned by the engine are deleted.

   Note: When you remove a messaging engine, WebSphere Application Server does not delete the data store tables automatically. You must remove them manually, or delete all the rows in all the tables. If you do this, a new messaging engine might fail to start if it attempts to use an orphaned data store. Refer to the documentation for your chosen relational database management system for information about deleting tables.

   Alternatively, for Apache Derby, you can delete the database directory, which is located in *profile_root*/databases/com.ibm.ws.sib, where *profile_root* is the directory in which profile-specific information is stored. However, do this only if the messaging engine is the sole user of the database.

   For more information, see Data store life cycle.

   Similarly, the file store files are not automatically deleted when you delete the messaging engine. You might want to delete the file store files to reclaim disk space.

Listing the messaging engines defined for a server bus member:

You can display a list of messaging engines defined for a server bus member by using the administrative console. You can decide which messaging engines you want to change, for example, which buses they are associated with.

About this task

To display the list of messaging engines, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click either **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines** or **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engines**. A list of messaging engines is displayed in the content pane.
2. Optional: Select one or more messaging engines to work with, for example to change the properties of the messaging engine.

Creating the database, schema and user ID for a messaging engine:

Before the data store for a messaging engine can be set up, you must first create the database, the schema and the database user ID that the messaging engine needs to access the data store tables.

Before you begin

Before you start this task, review the information in Configuration planning for a messaging engine to use a data store, and ensure that you have taken any appropriate action.

About this task

To create the database, schema and user ID for a messaging engine, complete the following steps.

Procedure

1. Create the database for the data store.
2. Create users and schemas in the database. Ensure that the user ID has sufficient privilege to allow the messaging engine to access the data store tables. For more information about the privileges that are required for the selected database, see “Database privileges” on page 1967.
3. If required, create the data store tables by using the data definition language (DDL) statements generated by using the sibDDLGenerator command.

Configuring a messaging engine data store to use a data source:

After configuring a JDBC data source, you can configure a messaging engine data store to use the data source.

Before you begin

To complete this task, you must have chosen or created a bus and a messaging engine, and the messaging engine must specify data store as its message store type.

You must also have configured a data source, as described in “Creating the database, schema and user ID for a messaging engine.”

About this task

A messaging engine uses an instance of a JDBC data source to interact with the database that contains the data store for that messaging engine.

Use the WebSphere Application Server administrative console to set the data store configuration parameters.

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name***.

2. Check that the **Message store type** is *Data store*.
3. Click **[Additional Properties] Message store**. The data store configuration detail panel is displayed.
4. Specify the following data store configuration details:

Data source JNDI name
Type the JNDI name of the data source that provides access to database that holds the data store.

Schema name
Type the name of the database schema that contains the tables used by the data store.

General tip: The schema name is usually the same as the user ID that is declared in the authentication alias. With some databases, for example DB2, you can provide an alternative schema name. For more information about the relationship between users and schema, refer to the documentation for your chosen RDBMS.

Informix tip: When you configure your messaging engine to use an Informix database, you must specify the schema name in lowercase letters.

When it is starting, a messaging engine that uses a data store checks to see if its data store exists. If the **Create tables** option is selected for the configuration, the messaging engine creates the tables in its chosen schema.

The **Schema name** field is optional. If you require a schema name, consider the following:

- The default schema name is `IBMWSSIB`.
- If you delete the text so that field is blank, the messaging engine takes the user id defined in the authentication alias to be the schema name.
- If you define a schema name explicitly, that schema name is used by the messaging engine.
- If there are multiple messaging engines, you must configure each messaging engine to use a unique schema, otherwise FFDC error messages stating that `Connection cannot be provided as Datasource has been disabled!` might appear. This applies to DB2 in particular.

Authentication alias
Select the authentication alias that enables access to the data source.

Apache Derby Tip: When you create a new Network Attached Apache Derby data store, by default you get a blank authentication alias.

Create tables
Select the check box if you want WebSphere Application Server to create the database tables automatically.

Note: The user ID that the messaging engine uses to connect to the data source must have sufficient authority to create the database tables and indexes.

DB2 for z/OS restriction: Do not select **Create tables** if you are using DB2 for z/OS, otherwise an exception will be thrown when WebSphere Application Server attempts to create the tables.

Number of tables for permanent objects
Permanent tables contain persistent objects for the data store.

Note: You can only increase the number of permanent tables, not decrease them.

Number of tables for temporary objects
Temporary tables contain nonpersistent objects that have been saved to the data store to reduce the messaging engine memory requirement.

Note: You can only increase the number of temporary tables, not decrease them.

Configuring service integration bus links:

You can configure service integration bus links on messaging engines in a variety of ways. For example, you can start, stop, or remove links.

About this task

When you create a foreign bus connection to connect two service integration buses, a service integration bus link is created automatically. The foreign bus connection contains a routing definition, which is a virtual link, and the service integration bus link is the corresponding physical link on the messaging engine.

Configuring foreign bus connections:

You can configure a bus to connect to, and exchange messages with, other messaging networks. To do this, you must configure a foreign bus connection. A foreign bus connection encapsulates information related to the remote messaging network, such as the type of the foreign bus and whether messaging applications are allowed to send messages to the foreign bus.

About this task

A foreign bus connection is associated with a service integration bus. The bus with which the foreign bus connection is associated is known as the local bus. The foreign bus connection represents another service integration bus, either in the same cell as the local bus or in a different cell, or it represents a WebSphere MQ queue manager. From the local bus, every other bus is regarded as a foreign bus, even if it is a bus in the same cell.

Messages route to a foreign bus either directly through a link between the local bus and the foreign bus, or indirectly through one or more intermediate buses.

If, when you configure the local bus, you select **Configuration reload enabled**, future changes to the configuration information for any foreign bus connections are updated automatically. The time when these changes take effect varies:

Foreign Bus Connection properties
: Immediately

WebSphere MQ link properties
: On channel restart, except Description (immediately), and Initial State (on messaging engine restart)

MQ sender channel properties
: On channel restart, except Initial State (on messaging engine restart or sender channel creation)

MQ receiver channel properties
: On channel restart, except Initial State (on messaging engine restart or receiver channel creation)

Publish/subscribe broker profile (0 to n) properties
: Immediately

Service integration bus link properties
: On link restart, except Description (immediately), and Initial State (on messaging engine restart or link creation)

To ensure that dynamic configuration updates are made to each node, click **System administration -> Console Preferences** to display the Console Preferences window, then select **Synchronize changes with nodes**.

Configuring the properties of a service integration bus link:

After establishing a service integration bus link you might want to configure the properties of a service integration bus link such as the name of the service integration bus link, or authentication alias used by foreign bus that the link connects to.

About this task

To configure the properties of a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine for which you want to configure the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to configure.
5. Specify the following properties for the service integration bus link:

   Name The name of the service integration bus link. In order to work, the name must be the same as the name of the corresponding service integration bus link configured on the target foreign bus.

   Description
   An optional description for the service integration bus link, for administrative purposes.

   UUID The universal unique identifier assigned by the system to the service integration bus link for administrative purposes.

   Foreign messaging engine
   The messaging engine on the foreign bus to which this link connects.

   Note: This foreign bus name must not be altered after it has been configured. If you alter it, any messaging engines that already hold state information on the link will not be able to use the link unless the foreign bus name is reset to its original value.

   Target inbound transport chain
   The type of transport chain used for communication with the foreign bus. The transport chain name must be the name of the transport chain as defined on the server on which the target messaging engine is hosted.

   Bootstrap endpoints
   The comma-separated list of endpoints used to connect to a bootstrap server. This property is set in the same way as the **Provider endpoint** property in the JMS connection factory settings. For more information, see the steps relating to setting bootstrap endpoints in “Configuring a connection to a non-default bootstrap server” on page 531.

   Note: Service integration bus links over BootstrapTunneledMessaging and BootstrapTunneledSecureMessaging transport chains only work directly between application server instances. Bus links over TunneledMessaging transport chains do not work if an HTTP server is placed in front of either application server instance.

   Authentication alias
   The name of the authentication alias, used to authenticate access to the foreign bus. The alias must be known to the foreign bus.

   Initial state
   Whether the link is started automatically when the messaging engine is started. Until started,

the gateway link is unavailable. If this property is set to **Started** the service integration bus link is started when the messaging engine is started.

6. Click **OK**.
7. Save your changes to the master configuration.

Listing the service integration bus links:

You can list all the service integration bus links that are linked to a messaging engine.

About this task

To list the service integration bus links for a messaging engine, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that your messaging engine belongs to.
3. In the content pane, under **Additional properties**, click **Messaging engines**. The list of messaging engines in the bus is displayed.
4. In the content pane, select the messaging engine for which you want to list the service integration bus links.
5. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.

Example

The following combinations of **Status** and **Activity** values are possible:

Table 205. Results of possible status and activity values. The first and second columns of the table provide the possible combinations of status and activity values, and the third column explains the possible connection status of the local and foreign buses.

Status	Activity	Meaning
started	inactive	The service integration bus link is started on the local messaging engine but has no connection to the foreign bus. The service integration bus link is attempting to activate a connection to the foreign bus. The service integration bus link on the foreign bus must also be started to successfully activate of a connection between the buses.
started	active	The service integration bus link is started on the local messaging engine and has an active connection to the foreign bus.
stopped	inactive	The service integration bus link is stopped on the local messaging engine and there is no connection to the foreign bus.
unknown	inactive	An error might have occurred in setting up the link, such that the object that is used to report the current state is not available.

What to do next

You can now add or remove a service integration bus link or select a service integration bus link to start, stop, or configure.

Starting a service integration bus link:

When a service integration bus link has been started, it can be used for communicating with its associated messaging engines.

About this task

To start a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine for which you want to start the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to start.
5. Click **Start**.

Stopping a service integration bus link:

When a service integration bus link has been stopped, it cannot be used for communication until it is restarted.

About this task

To stop a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine to which the service integration bus link that you want to stop belongs.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to stop.
5. Click **Stop**.

Removing a service integration bus link:

When a service integration bus link to a messaging engine has been removed, it cannot be used for communication until it is recreated.

Before you begin

Before you remove the service integration bus link you must stop the link.

When you remove a service integration bus link, all traffic that uses the link needs to be dealt with in a similar way to when you remove a destination. For further details, see “Deleting a non-topic space bus destination” on page 1997. Note that express Quality of Service (QoS) messages are discarded.

About this task

To remove a service integration bus link from a messaging engine, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine that is associated with the service integration bus link that you want to remove.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to remove.
5. Click **Delete**.
6. Save your changes to the master configuration.

What to do next

When you remove a service integration bus link, all traffic that uses the link must be diverted by alternative routes or held pending further administrative action. Express QoS messages are discarded.

Configuring bus destinations

Use the following tasks to configure permanent bus destinations on service integration buses.

About this task

The steps involved in configuring a bus destination depend on the intended usage of the destination.

For example, the following figure shows a basic scenario based on an application using a JMS queue for point-to-point messaging. A producing application sends messages to a JMS queue from which a consuming application retrieves the messages. The JMS queue is assigned to a queue destination and its associated queue point, where messages are stored.

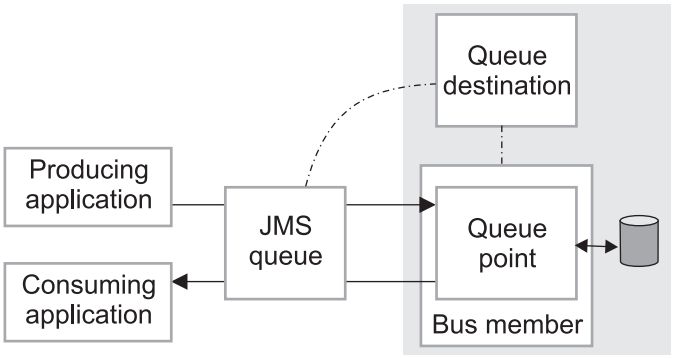


Figure 36. Application use of destinations - basic case

To enable the JMS applications to use a queue destination, you configure the following administrative destination objects:

1. A queue destination on a service integration bus. This configures the properties of the queue, such as the name, and associates the queue with one bus member (an application server). This also automatically configures, on the bus member, a queue point where messages for the queue are held and processed.
2. A JMS queue, which configures the name that applications can use to look up the queue in JNDI. The JMS queue encapsulates the name of the queue destination, as defined in the queue destination above, together with other properties to be used by applications.

After you have created a queue destination, you can optionally configure the queue point to override some properties configured on the queue destination. You can also undertake other configuration tasks on the destination and its queue point, and can act on the runtime view.

Each messaging engine has a default exception destination, named `_SYSTEM.Exception.Destinaton.`*`messaging_engine_name`*. This exception destination can be used to handle messages that cannot be delivered for all bus destinations that are localized to the messaging engine. Each bus destination can be configured with a non-default exception destination.

For more information about configuring bus destinations, see the following topics:
- Listing bus destinations
- Creating a bus destination
- Configuring qualities of service for a destination
- Deleting a non-topic space bus destination
- Deleting a topic space

Connecting buses

You can connect several service integration buses in a network directly, or indirectly, by creating and configuring foreign bus connections. You can connect a service integration bus and a WebSphere MQ queue manager or queue-sharing group in a similar way. You can enable point-to-point or publish/subscribe messaging across multiple buses.

About this task

For a conceptual overview of connecting different service integration buses, including security issues, see Interconnected buses. You can configure a bus to connect to, and exchange messages with, other messaging networks. To do this, you must configure a foreign bus connection. A foreign bus connection encapsulates information related to the remote messaging network, such as the type of the foreign bus and whether messaging applications are allowed to send messages to the foreign bus. Messages route to a foreign bus either directly through a link between the local bus and the foreign bus, or indirectly through one or more intermediate buses.

Configuring foreign bus connections:

You can configure a bus to connect to, and exchange messages with, other messaging networks. To do this, you must configure a foreign bus connection. A foreign bus connection encapsulates information related to the remote messaging network, such as the type of the foreign bus and whether messaging applications are allowed to send messages to the foreign bus.

About this task

A foreign bus connection is associated with a service integration bus. The bus with which the foreign bus connection is associated is known as the local bus. The foreign bus connection represents another service integration bus, either in the same cell as the local bus or in a different cell, or it represents a WebSphere MQ queue manager. From the local bus, every other bus is regarded as a foreign bus, even if it is a bus in the same cell.

Messages route to a foreign bus either directly through a link between the local bus and the foreign bus, or indirectly through one or more intermediate buses.

If, when you configure the local bus, you select **Configuration reload enabled**, future changes to the configuration information for any foreign bus connections are updated automatically. The time when these changes take effect varies:

Foreign Bus Connection properties
: Immediately

WebSphere MQ link properties
On channel restart, except Description (immediately), and Initial State (on messaging engine restart)

MQ sender channel properties
On channel restart, except Initial State (on messaging engine restart or sender channel creation)

MQ receiver channel properties
On channel restart, except Initial State (on messaging engine restart or receiver channel creation)

Publish/subscribe broker profile (0 to n) properties
Immediately

Service integration bus link properties
On link restart, except Description (immediately), and Initial State (on messaging engine restart or link creation)

To ensure that dynamic configuration updates are made to each node, click **System administration -> Console Preferences** to display the Console Preferences window, then select **Synchronize changes with nodes**.

Connecting a bus and a WebSphere MQ gateway queue manager to use point-to-point messaging:

You can connect a service integration bus to a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue sharing group to send or receive messages by using point-to-point messaging. One way to do this is to create a foreign bus connection, where the WebSphere MQ queue manager or queue-sharing group is configured as a foreign bus.

Before you begin

To connect a service integration bus and a WebSphere MQ queue manager or queue-sharing group to use point-to-point messaging, the following resources must be defined in WebSphere Application Server:

- A service integration bus that you want to connect from, known as the local bus. The bus must have at least one bus member.

The following resources must be defined in WebSphere MQ:

- A queue manager or queue-sharing group, which acts as the gateway to the WebSphere MQ network.
- A listener that is configured and running.
- A sender channel (to receive messages on the local bus), a receiver channel (to send messages from the local bus), or both.

About this task

In point-to-point messaging, the sending application specifies the target destination for the message. To receive the message, the receiving application specifies the same destination when it communicates with the messaging provider. Therefore, there is a one-to-one mapping between the sender and receiver of a message.

This task describes one way to achieve point-to-point messaging between WebSphere MQ queue manager or queue-sharing group. For further information about interoperation with a WebSphere MQ network, see the related tasks.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that you want to connect from, that is, the local bus.

3. In the configuration tab, under **Topology**, click **Foreign bus connections**.
4. In the Foreign bus connections pane, click **New** to start the Foreign bus connection wizard.
5. In the Bus connection type pane, ensure that **Direct connection** is selected.
6. In the Foreign bus type pane, select **WebSphere MQ**.
7. In the Local bus details pane, select the messaging engine that you want to use and enter the name of the virtual queue manager, that is, the name by which the virtual queue manager of the service integration bus is known to the WebSphere MQ network.
8. In the WebSphere MQ details pane, enter a name for the foreign bus, that is, the bus that represents the WebSphere MQ queue manager. Enter a name for the WebSphere MQ link that connects to the foreign bus. Ensure that these two names are not the same.
9. Ensure that the **Configure publish-subscribe messaging for this connection** check box is clear.
10. To send messages from the local bus to the WebSphere MQ queue manager, complete the following details:
    a. Ensure that **Enable Service integration bus to WebSphere MQ message flow** is selected.
    b. Enter the WebSphere MQ receiver channel name, host name and communication port.
    c. If the WebSphere MQ queue manager requires a secure connection, select the **Is the WebSphere MQ receiver channel secure?** check box. When this option is selected, the WebSphere MQ receiver channel accepts only connections that have secure sockets layer (SSL) based encryption. The connection is successful only if a set of suitably compatible SSL credentials are associated with the service integration bus outbound channel and the WebSphere MQ receiver channel that it connects to.
11. To receive messages on the local bus from the WebSphere MQ queue manager, complete the following details:
    a. Ensure that **Enable WebSphere MQ to Service integration bus message flow** is selected.
    b. Enter the WebSphere MQ sender channel name.
    c. Optionally, enter the service integration bus inbound user ID. When the local bus is secure, the inbound user ID replaces the user ID in messages from the foreign bus that arrive at the local bus and is used to authorize whether those messages can access their destinations. Specify an inbound user ID for the local service integration bus under the following circumstances:
       - The foreign bus is in a different security domain, so user IDs in the foreign bus are not recognized in the local bus.
       - You want local control of access to inbound messages to the local bus.

       If the local bus is not secure, the inbound user ID has no effect on messages. If the local bus is secure, the foreign bus is not secure, and an inbound user ID is not set, an inbound message from the foreign bus is only authorized to destinations that allow unauthenticated users access.
12. When the Foreign bus connection wizard is finished, save your changes to the master configuration.

Results

You have created a connection between a service integration bus and a WebSphere MQ queue manager to use point-to-point messaging. You have created a direct foreign bus connection, which contains a routing definition, or virtual link. The physical link, a WebSphere MQ link on the messaging engine for the local bus, is created automatically.

What to do next

You can test the connection.

Connecting a bus and a WebSphere MQ network to use publish/subscribe messaging:

You can connect a service integration bus and a WebSphere MQ network to send and receive messages by using publish/subscribe messaging. To do this, you create a foreign bus connection, where the WebSphere MQ network is viewed as a foreign bus.

Before you begin

To connect a service integration bus and a WebSphere MQ network to use publish/subscribe messaging, the following resources must be defined in WebSphere Application Server:

- A service integration bus that you want to connect from, known as the local bus. The bus must have at least one bus member.

The following resources must be defined in WebSphere MQ:

- A queue manager or (for WebSphere MQ for z/OS) queue-sharing group, which acts as the gateway to the WebSphere MQ network.
- A listener that is configured and running.
- A topic and input queue for broker publish/subscribe flow configured in WebSphere MQ.
- A sender channel (to receive messages on the local bus), a receiver channel (to send messages from the local bus), or both.

About this task

In publish/subscribe messaging, the sending application publishes messages to an intermediate broker destination. Multiple receiving applications can subscribe to this destination to receive a copy of any messages that are published. When a message arrives at a destination, the messaging provider distributes a copy of the message to all the receiving applications that subscribe to the destination. There can be a one-to-many relationship between the sender and receiver of a message, depending on how many receiving applications are subscribed to a destination when a message arrives.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that you want to connect from, that is, the local bus.
3. In the configuration tab, under **Topology**, click **Foreign bus connections**.
4. In the Foreign bus connections pane, click **New** to start the Foreign bus connection wizard.
5. In the Bus connection type pane, ensure that **Direct connection** is selected.
6. In the Foreign bus type pane, select **WebSphere MQ**.
7. In the Local bus details pane, select the messaging engine that you want to use and enter the name of the virtual queue manager, that is, the name by which the virtual queue manager of the service integration bus is known to the WebSphere MQ network.
8. In the WebSphere MQ details pane, complete the following details:
   a. Enter a name for the foreign bus, that is, the bus that represents the WebSphere MQ network.
   b. Enter a name for the WebSphere MQ link that connects to the foreign bus. Ensure that the Foreign bus name and MQ link name are different.
   c. Select the **Configure publish-subscribe messaging for this connection** check box.
9. To send messages from the local bus to the WebSphere MQ network, complete the following details:
   a. Ensure that **Enable Service integration bus to WebSphere MQ message flow** is selected.
   b. Enter the WebSphere MQ receiver channel name, host name and communication port.
   c. If the WebSphere MQ gateway queue manager or queue-sharing group requires a secure connection, select the **Is the WebSphere MQ receiver channel secure?** check box. When this option is selected, the WebSphere MQ receiver channel accepts only connections that have

secure sockets layer (SSL) based encryption. The connection is successful only if a set of suitably compatible SSL credentials are associated with the service integration bus outbound channel and the WebSphere MQ receiver channel that it connects to.

10. To receive messages on the local bus from the WebSphere MQ network, complete the following details:
    a. Ensure that **Enable WebSphere MQ to Service integration bus message flow** is selected.
    b. Enter the WebSphere MQ sender channel name.
    c. Optionally, enter the service integration bus inbound user ID. When the local bus is secure, the inbound user ID replaces the user ID in messages from the foreign bus that arrive at the local bus and is used to authorize whether those messages can access their destinations. Specify an inbound user ID for the local service integration bus under the following circumstances:
       - The foreign bus is in a different security domain, so user IDs in the foreign bus are not recognized in the local bus.
       - You want local control of access to inbound messages to the local bus.

       If the local bus is not secure, the inbound user ID has no effect on messages. If the local bus is secure, the foreign bus is not secure, and an inbound user ID is not set, an inbound message from the foreign bus is only authorized to destinations that allow unauthenticated users access.
11. From the Publish-subscribe details pane, repeat the following steps for each topic mapping you want to create:
    a. Enter the name of the topic on the local bus.
    b. Select the name of the topic space on the local bus that will map to the topic space on the foreign bus.
    c. Enter the name of the gateway queue manager or queue sharing group for the WebSphere MQ broker configured for broker publish/subscribe flow.
    d. To send messages from the local bus to the WebSphere MQ gateway queue manager or queue-sharing group, enter the name of the queue for the WebSphere MQ broker destination.
    e. To receive messages on the local bus from the WebSphere MQ gateway queue manager or queue-sharing group, enter the name of the subscription point that will receive messages.
    f. Select the direction of message flow for the publish/subscribe topic mapping. The options available depend on whether you completed details on the WebSphere MQ details pane to send messages, receive messages, or both, on the local bus.
    g. Click **Add**.
12. When the Foreign bus connection wizard is finished, save your changes to the master configuration.

Results

You have created a connection between a service integration bus and a WebSphere MQ network to use publish/subscribe messaging. You have created a direct foreign bus connection, which contains a routing definition, or virtual link. The physical link, a WebSphere MQ link on the messaging engine for the local bus, is created automatically.

What to do next

You can test the connection.

Connecting service integration buses to use point-to-point messaging:

You can connect a service integration bus to another service integration bus to send and receive messages by using point-to-point messaging. To do this, you create a foreign bus connection.

Before you begin

To connect a service integration bus to another service integration bus to use point-to-point messaging, the following resources must be defined:

- A service integration bus that you want to connect from, known as the local bus. The bus must have at least one bus member.
- A service integration bus that you want to connect to, known as the foreign bus. The bus must have at least one bus member.
- Optionally, to configure a secure connection, an authentication alias.

The buses that you connect must have unique names, because the connection will fail if the buses have the same name.

About this task

In point-to-point messaging, the sending application specifies the target destination for the message. To receive the message, the receiving application specifies the same destination when it communicates with the messaging provider. Therefore, there is a one-to-one mapping between the sender and receiver of a message.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that you want to connect from, that is, the local bus.
3. In the configuration tab, under **Topology**, click **Foreign bus connections**.
4. In the Foreign bus connections pane, click **New** to start the Foreign bus connection wizard.
5. In the Bus connection type pane, ensure that **Direct connection** is selected.
6. In the Foreign bus type pane, ensure that **Service integration bus** is selected.
7. In the Local bus details pane, select from the drop-down list the messaging engine that you want to use.
8. Optionally, enter a name for the inbound user ID. When the local bus is secure, the inbound user ID replaces the user ID in messages from the foreign bus that arrive at the local bus and is used to authorize whether those messages can access their destinations. Specify an inbound user ID for the local service integration bus under the following circumstances:
   - The foreign bus is in a different security domain, so user IDs in the foreign bus are not recognized in the local bus.
   - You want local control of access to inbound messages to the local bus.

   If the local bus is not secure, the inbound user ID has no effect on messages. If the local bus is secure, the foreign bus is not secure, and an inbound user ID is not set, an inbound message from the foreign bus is only authorized to destinations that allow unauthenticated users access.
9. In the Foreign bus details pane, complete the details as appropriate:
   - If the service integration bus you want to connect to is in a different cell from the local bus, complete the following details:
     a. Ensure that **Configure a foreign bus in a remote cell** is selected.
     b. Enter the name of the service integration bus to connect to, that is, the foreign bus. Enter the exact name of the existing service integration bus.
     c. Enter the name of the gateway messaging engine in the foreign bus, that is, the messaging engine to connect to in the foreign bus.
     d. Ensure that the **Configure publish-subscribe messaging for this connection** check box is clear.

e. Enter the name of the service integration bus link.

f. Enter one or more bootstrap endpoints, that is, the host, port location, and transport chain for the messaging engine on the foreign bus that the local service integration bus connects to. The port is the SIB_ENDPOINT_ADDRESS (or SIB_ENDPOINT_SECURE_ADDRESS if security is enabled) of the messaging engine. Use the format *hostName*:*portNumber*:*chainName*, separating each bootstrap endpoint by a comma. For more information, see the steps relating to setting bootstrap endpoints in "Configuring a connection to a non-default bootstrap server" on page 531.

- If the service integration bus you want to connect to is in the same cell as the local bus, complete the following details:

  a. Ensure that **Configure a foreign bus in a local cell** is selected.

  b. Select the name of the service integration bus to connect to, that is, the foreign bus.

  c. Select the name of the gateway messaging engine in the foreign bus, that is, the messaging engine to connect to in the foreign bus.

  d. Ensure that the **Configure publish-subscribe messaging for this connection** check box is clear.

  e. Enter the name of the service integration bus link.

10. Optionally, to secure the connection, in the Foreign bus details pane, complete the following details:

    a. Select the **Secure connection** check box.

    b. Select the type of transport chain to use to communicate with the messaging engine in the foreign bus. Select one of the following:

       - InboundBasicMessaging. InboundBasicMessaging is a predefined transport chain where communication uses the TCP protocol.
       - InboundSecureMessaging. InboundSecureMessaging is a predefined transport chain where communication is secured by using the secure sockets layer (SSL) based encryption protocol over a TCP network. For successful connection, a set of suitably compatible SSL credentials must be associated with the local bus inbound channel and the foreign bus outbound channel.
       - Other, please specify. Select this option to specify your own transport chain and enter the details in the field that appears.

    c. Select the name of the authentication alias to use to authenticate access to the foreign bus. The alias must be known to the foreign bus.

11. When the Foreign bus connection wizard is finished, save your changes to the master configuration.

Results

You have created a connection from a local service integration bus to a foreign service integration bus to use point-to-point messaging. You have created a direct foreign bus connection, which contains a routing definition, or virtual link. The physical link, a service integration bus link on the messaging engine for the local bus, is created automatically.

What to do next

You must create a connection in the opposite direction between the two buses. To do this, repeat the procedure, using the bus you have just connected to as the local bus, and the bus you have just connected from as the foreign bus. Ensure that you use exactly the same name for the service integration bus link. After you create a foreign bus connection for each service integration bus, you can test the connection.

Connecting service integration buses to use publish/subscribe messaging:

You can connect a service integration bus to another service integration bus to send and receive messages by using publish/subscribe messaging. To do this, you create a foreign bus connection.

Before you begin

To connect a service integration bus to another service integration bus to use publish/subscribe messaging, the following resources must exist:

- A service integration bus that you want to connect from, known as the local bus. The bus must have at least one bus member.
- A service integration bus that you want to connect to, known as the foreign bus. The bus must have at least one bus member.
- A topic space on both service integration buses. If the foreign bus is in a remote cell, you must know the topic space name.
- Optionally, to configure a secure connection, an authentication alias.

The buses that you connect must have unique names, because the connection will fail if the buses have the same name.

About this task

In publish/subscribe messaging, the sending application publishes messages to an intermediate broker destination. Multiple receiving applications can subscribe to this destination to receive a copy of any messages that are published. When a message arrives at a destination, the messaging provider distributes a copy of the message to all the receiving applications that subscribe to the destination. There can be a one-to-many relationship between the sender and receiver of a message, depending on how many receiving applications are subscribed to a destination when a message arrives.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that you want to connect from, that is, the local bus.
3. In the configuration tab, under **Topology**, click **Foreign bus connections**.
4. In the Foreign bus connections pane, click **New** to start the Foreign bus connection wizard.
5. In the Bus connection type pane, ensure that **Direct connection** is selected.
6. In the Foreign bus type pane, ensure that **Service integration bus** is selected.
7. In the Local bus details pane, select from the drop-down list the messaging engine that you want to use.
8. Optionally, enter a name for the inbound user ID. When the local bus is secure, the inbound user ID replaces the user ID in messages from the foreign bus that arrive at the local bus and is used to authorize whether those messages can access their destinations. Specify an inbound user ID for the local service integration bus under the following circumstances:
   - The foreign bus is in a different security domain, so user IDs in the foreign bus are not recognized in the local bus.
   - You want local control of access to inbound messages to the local bus.

   If the local bus is not secure, the inbound user ID has no effect on messages. If the local bus is secure, the foreign bus is not secure, and an inbound user ID is not set, an inbound message from the foreign bus is only authorized to destinations that allow unauthenticated users access.
9. In the Foreign bus details pane, complete the details as appropriate:
   - If the service integration bus you want to connect to is in a different cell from the local bus, complete the following details:
     a. Ensure that **Configure a foreign bus in a remote cell** is selected.
     b. Enter the name of the service integration bus to connect to, that is, the foreign bus. Enter the exact name of the existing service integration bus.

c. Enter the name of the gateway messaging engine in the foreign bus, that is, the messaging engine to connect to in the foreign bus.
d. Select the **Configure publish-subscribe messaging for this connection** check box.
e. Enter the name of the service integration bus link.
f. Enter one or more bootstrap endpoints, that is, the host, port location, and transport chain for the messaging engine on the foreign bus that the local service integration bus connects to. The port is the SIB_ENDPOINT_ADDRESS (or SIB_ENDPOINT_SECURE_ADDRESS if security is enabled) of the messaging engine. Use the format *hostName*:*portNumber*:*chainName*, separating each bootstrap endpoint by a comma. For more information, see the steps relating to setting bootstrap endpoints in "Configuring a connection to a non-default bootstrap server" on page 531.

- If the service integration bus you want to connect to is in the same cell as the local bus, complete the following details:
  a. Ensure that **Configure a foreign bus in a local cell** is selected.
  b. Select the name of the service integration bus to connect to, that is, the foreign bus.
  c. Select the name of the gateway messaging engine in the foreign bus, that is, the messaging engine to connect to in the foreign bus.
  d. Select the **Configure publish-subscribe messaging for this connection** check box.
  e. Enter the name of the service integration bus link.

10. Optionally, to secure the connection, in the Foreign bus details pane, complete the following details:
    a. Select the **Secure connection** check box.
    b. Select the type of transport chain to use to communicate with the messaging engine in the foreign bus. Select one of the following:
       - InboundBasicMessaging. InboundBasicMessaging is a predefined transport chain where communication uses the TCP protocol.
       - InboundSecureMessaging. InboundSecureMessaging is a predefined transport chain where communication is secured by using the secure sockets layer (SSL) based encryption protocol over a TCP network. For successful connection, a set of suitably compatible SSL credentials must be associated with the local bus inbound channel and the foreign bus outbound channel.
       - Other, please specify. Select this option to specify your own transport chain and enter the details in the field that appears.
    c. Select the name of the authentication alias to use to authenticate access to the foreign bus. The alias must be known to the foreign bus.
11. In the Publish-subscribe details pane, repeat the following steps for each topic mapping you want to create:
    a. Select the name of the topic space on the local bus that will map to the topic space on the foreign bus.
    b. Enter the name of the topic space on the foreign bus. If the foreign bus is in the same cell as the local bus, you can select this name from a drop-down list.
    c. Click **Add**.
12. When the Foreign bus connection wizard is finished, save your changes to the master configuration.

Results

You have created a connection from a local service integration bus to a foreign service integration bus to use publish/subscribe messaging. You have created a direct foreign bus connection, which contains a routing definition, or virtual link. The physical link, a service integration bus link on the messaging engine for the local bus, is created automatically.

What to do next

You must create a connection in the opposite direction between the two buses. To do this, repeat the procedure, using the bus you have just connected to as the local bus, and the bus you have just connected from as the foreign bus. Ensure that you use exactly the same name for the service integration bus link. After you create a foreign bus connection for each service integration bus, you can test the connection.

Connecting buses by using an indirect connection:

You can connect a service integration bus to another service integration bus or a WebSphere MQ queue manager or queue-sharing group (known as the “gateway queue manager”) through one or more intermediate foreign buses. To do this, you create an indirect foreign bus connection.

Before you begin

To create an indirect foreign bus connection, the following resources must be defined:

- A service integration bus that you want to connect from, known as the local bus.
- A direct foreign bus connection from the local bus to the bus you want to use as the intermediate bus.
- A service integration bus or a gateway queue manager that you want to connect to indirectly, known as the target foreign bus.

About this task

To connect two buses indirectly, you create a indirect foreign bus connection on the local bus that specifies a suitable intermediate bus and the required target bus. There must be a connection, either direct or indirect, between the intermediate bus and the target foreign bus, that is, the service integration bus or a gateway queue manager that you want to connect to.

To connect one bus to another bus through an intermediate bus, or a chain of buses, if the connection between the intermediate bus or the chain of buses and the target bus already exists, you do not require any new physical links. Instead, each foreign bus connection identifies a neighboring bus on the route to the final target bus as the "next hop" in the chain. Each bus in the chain must know about the next hop in the chain to reach the target bus. The local bus uses a foreign bus connection to identify the next bus in the chain to the target bus, and uses its direct physical link to flow messages to that bus. Then, each intermediate bus uses its locally defined foreign bus connection to identify the next bus in the chain until the target bus is reached.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that you want to connect from, that is, the local bus.
3. In the configuration tab, under **Topology**, click **Foreign bus connections**.
4. In the Foreign bus connections pane, click **New** to start the Foreign bus connection wizard.
5. In the Bus connection type pane, select **Indirect, using another bus**.
6. In the Indirect connection details pane, complete the following details:
   a. Enter the name of the target foreign bus that you want to connect to indirectly.
   b. Select the name of the existing, directly connected foreign bus that will be the intermediate bus to the target bus.
7. When the Foreign bus connection wizard is finished, save your changes to the master configuration.

Results

You have created a connection from the local bus to the target bus by way of the intermediate bus. You have created an indirect foreign bus connection, which contains a routing definition, or virtual link.

What to do next

Check whether there are connections between the intermediate bus you specified and the target bus. If necessary, repeat the procedure to create the "next hop" in the chain.

Testing foreign bus connections:

After you define a direct foreign bus connection between a service integration bus and either another service integration bus or a WebSphere MQ queue manager, you can test the connection to check that it can be used to exchange messages.

Before you begin

To test a foreign bus connection, a service integration bus, known as the local bus, with a direct foreign bus connection to a service integration bus, or a WebSphere MQ queue manager must be defined.

About this task

You might want to test a foreign bus connection before you send messages that use that connection, or if you have difficulty sending and receiving messages that use that connection. You can test only direct foreign bus connections. You cannot test indirect foreign bus connections, because it might not be possible to reach the target destination bus.

If a connection name list is provided in the WebSphere MQ link sender channel properties, the connections are tried in the order in which they are specified in the connection list until a connection is successful. If no connection is successful an error message is logged.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the Buses pane, click the bus that has the connection you want to test.
3. In the configuration tab, under **Topology**, click **Foreign bus connections**. A list of foreign bus connections is displayed.
4. Select the check box for the connection you want to test.
5. Click **Test Connection**. A message is displayed at the top of the pane that shows the result of the test.

Results

You have tested a bus connection.

Listing the foreign bus connections:

You can list the foreign bus connections that are associated with a local bus to see which foreign buses are currently configured.

Before you begin

To list foreign bus connections, the following resources must be defined:

- A service integration bus with one or more foreign bus connections between it and another service integration bus or a WebSphere MQ queue manager.

About this task

You can list foreign bus connections by using the administrative console, as described in this topic, or by using the wsadmin tool and the listSIBForeignBuses command.

To list the foreign bus connections that are associated with the local bus, complete the following steps.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses**.
3. In the service integration buses pane, select the bus for which you want to list the connections.
4. In the configuration tab, under **Topology**, click **Foreign bus connections**. A list of buses that have a foreign bus connection from the local bus is displayed.

Results

You have listed the foreign bus connections that are associated with the local bus.

Removing a foreign bus connection from a bus:

You can remove a foreign bus connection from a bus so that the connection can no longer be used for sending and receiving messages.

Before you begin

When a foreign bus connection is deleted from the configuration, the next time the hosting messaging engine for a link transmitter is started, the messaging engine moves the messages to the exception destination. To avoid this situation, drain the messages from the link transmitter queue as far as is possible and then, before deleting the link configuration, either move any remaining messages from the transmission queues to an exception destination, or delete them. For more details, refer to the subtopics that follow.

About this task

Removing a foreign bus connection from a bus removes the resources associated with the connection. The bus will no longer service the deleted link.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the Buses pane, select the bus from which you want to remove the foreign connection.
3. In the configuration tab, under **Topology**, click **Foreign bus connections** to display a list of connections for this bus.
4. Select the check box that corresponds to the connection that you want to remove.
5. Click **Delete**.
6. Save your changes to the master configuration.

Results

You have now deleted the foreign bus connection from a bus. This connection can no longer be used for communication.

Preparing to remove a foreign bus connection between two service integration buses:

Before you remove a foreign bus connection between two service integration buses, drain as many messages as you can from the link transmitter queue, then manually move or delete any remaining messages.

Before you begin

You must know which foreign bus connection is being prepared for deletion.

About this task

When a foreign bus connection and its service integration bus link is deleted from the configuration, the next time the hosting messaging engine for a link transmitter is started, the messaging engine deletes all its messages or moves them to the exception destination. To avoid messages being unintentionally deleted or moved to the exception destination, drain the messages from the link transmitter queue as much as possible and then, before deleting the link configuration, either move any remaining messages to an exception destination or delete them.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses** to display a list of buses.
3. Select the bus from which you want to remove the foreign bus connection.
4. In the **Configuration** tab, under **Topology**, click **Foreign bus connections**. A list of connections for this bus is displayed.
5. From the list of foreign bus connections, click the name of the foreign bus connection to display its details.
6. Under **General properties**, clear the **Send Allowed** check box to prevent new messages from being produced for this service integration bus link.
7. Click **Apply** to save the configuration.
8. To determine when there are no more messages queued on this link, under **Related Items** click **Service integration bus links** to display the details of the service integration bus links.
9. Click the **Refresh** icon of **Status** to refresh the view of the current outbound messages.
10. When there are no current outbound messages, select the check box next to the appropriate link and then click **Stop** to stop the link to the foreign bus. When the status of the link turns to red, the link to the foreign bus has no remaining messages and is stopped.
11. Repeat steps 2 to 10 for the foreign bus, because the foreign bus can continue to produce messages after the foreign bus connection on the local bus has been deleted.
12. Save your changes to the master configuration.

Results

You have drained the messages from the link transmitter queue as much as possible, and either moved any remaining messages from the transmission queues to an exception destination or deleted them. You are now ready to remove the foreign bus connection.

Preparing to remove a foreign bus connection between a service integration bus and a WebSphere MQ network:

Before you remove a foreign bus connection between a service integration bus and a WebSphere MQ network, drain as many messages as you can from the link transmitter queue, then manually move or delete any remaining messages.

Before you begin

You must know which foreign bus connection is being prepared for deletion. You should also inform the WebSphere MQ administrator that the foreign bus connection is about to be deleted and therefore no longer paired with its WebSphere MQ gateway queue manager or message broker in the WebSphere MQ network.

About this task

When a foreign bus connection is deleted from the configuration, the next time the hosting messaging engine for a link transmitter is started, it deletes all its messages or moves them to the exception destination. To avoid messages being unintentionally deleted or moved to the exception destination, drain as many messages as possible from the link transmitter queue. Then, before you delete the link configuration, either move any remaining messages to an exception destination or delete them.

If there are publish/subscribe broker profiles defined, you should remove the subscriptions.

Procedure

1. Start the administrative console.
2. Optional: If there are publish/subscribe broker profiles defined on any of the links for this foreign bus connection, remove the subscriptions.

   Complete the following substeps for each broker profile:

   a. Navigate to **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name***
   b. Click the **Runtime** tab.
   c. Click **Subscriptions**.
   d. Click **Unsubscribe** to remove all the subscriptions listed.

   When an unsubscribe command is sent to the message broker in the WebSphere MQ network, the relevant topic mapping is put into an indoubt state until the unsubscribe is confirmed when the topic mapping is deleted. After the unsubscribe is confirmed, the topic mapping is no longer shown in the runtime view. You might have to refresh the runtime view for all subscriptions to be shown as removed.
3. Prevent new messages from being produced for this foreign bus connection.
   a. Navigate to **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name***
   b. Under **General properties**, clear the **Send Allowed** check box.
   c. Click **Apply** to save the configuration.
4. Determine when there are no more messages queued, then stop the link to the foreign bus in a controlled manner.
   a. Under **Related Items**, click **WebSphere MQ links** to display the list of links for this bus.
   b. Click the **Refresh** icon of **Status** to refresh the view of the current outbound messages.
   c. When there are no current outbound messages, select the check box next to the appropriate link and then select a **Stop mode** of "Quiesce".
   d. Select a **Target state** of "Stopped" so that the link can only be started again by administrator action.
   e. When the status of the link turns to red, the link to the foreign bus has no remaining messages and is stopped.
5. The foreign bus can continue to produce messages after the foreign bus connection on the local bus has been deleted. Because the foreign bus is a WebSphere MQ network, refer to the WebSphere MQ Intercommunication guide for details about the safe deletion of channels at Managing WebSphere MQ channels.

6. Save your changes to the master configuration.

Results

You have removed the subscriptions from any publish/subscribe brokers on the link. You have drained as many messages as possible from the link transmitter queue, and either moved any remaining messages from the transmission queues to an exception destination or deleted them.

What to do next

You are now ready to remove the foreign bus connection.

Configuring destination defaults for a foreign bus connection:

You can configure default values to apply when applications use destinations on a foreign bus and these properties are not explicitly defined elsewhere. You configure the destination defaults for the foreign bus connection, which represents either a service integration bus or a WebSphere MQ queue manager or queue-sharing group (known as a “gateway queue manager”).

Before you begin

To configure destination defaults for a foreign bus, at least one foreign bus connection that represents that foreign bus must be defined.

About this task

Destination defaults are default values for the destination attributes, such as the default priority. They are used when an application produces messages that are destined for a destination on a foreign bus, but the properties are not defined elsewhere:

- The application has not explicitly set certain attributes on the message.
- A foreign destination is not defined on the local bus, so cannot be a source of destination defaults.
- An alias destination is not defined on the local bus, so cannot be a source of destination defaults.

You configure destination defaults for a foreign bus connection to apply to all applicable destinations on the corresponding foreign bus. In contrast, the destination defaults of an alias destination or foreign destination apply to an individual destination on a foreign bus.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the service integration buses pane, select the bus that you want to configure.
3. In the configuration tab, under **Topology**, click **Foreign bus connections**. A list of foreign bus connections is displayed.
4. Select the foreign bus connection that you want to configure.
5. In the content pane, under **Additional properties**, click **Destination defaults**.
6. Specify the following properties for the destination defaults:

   Default priority
   The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

   Default reliability
   The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

Maximum reliability
The maximum reliability of messages accepted by this destination.

Send allowed
Clear this option (setting it to false) to stop producers from being able to send messages to destinations on this foreign bus.

Enable producers to override default reliability
Select this option to enable producers to override the default reliability that is set on the destination.

Include an RFH2 message header when sending messages to WebSphere MQ.
If selected, messages sent to WebSphere MQ include an RFH2 header. The RFH2 header stores additional information to that which is stored in theWebSphere MQ message header.

7. Click **OK**.
8. Save your changes to the master configuration.

Results

You have configured destination defaults for a foreign bus connection.

Managing messages that use foreign bus connections:

You can manage messages that use a foreign bus connection.

Before you begin

You must know whether the foreign bus connection represents a connection from a service integration bus to another service integration bus, or to a WebSphere MQ network, because this determines the points where you can manage messages.

About this task

For a foreign bus connection from a service integration bus to another service integration bus, you can manage messages that are queued for link transmitters and link receivers.

For a foreign bus connection from a service integration bus to a WebSphere MQ network, you can manage messages that are queued for link transmitters, known link transmitters, and the sender channel transmitter.

A link can have multiple link transmitters; one for each messaging engine that sends messages on the link to the foreign bus, and one for each topic space destination that is mapped to a topic space in the foreign bus.

For a foreign bus connection to a WebSphere MQ network, there is one known link transmitter for each link transmitter, and one sender channel transmitter for the WebSphere MQ link that sends messages to a gateway queue manager on the WebSphere MQ network.

Complete this task when you want to view, delete, or move messages on a transmitter.

Procedure

Select the appropriate task from the following topics.

Configuring exception destination processing for a link to a foreign bus:

You can configure the exception destination processing for a link from a service integration bus to another service integration bus or to a WebSphere MQ network. You can configure whether any undeliverable messages that the link handles are rerouted to an exception destination, and whether to use a system default exception destination or configure a specific exception destination.

Before you begin

To configure the exception destination processing for a service integration bus link or WebSphere MQ link, you must know the name of the foreign bus connection that is associated with that link. A foreign bus connection represents either a connection between two service integration buses or a connection between a service integration bus and a WebSphere MQ network.

To configure a specific exception destination for the link, the exception destination must exist. An exception destination must be a queue destination, and can be on the local bus or a foreign bus. See “Creating a queue for point-to-point messaging” on page 1972.

About this task

An exception destination for a link is the destination for an inbound message when the service integration bus link or WebSphere MQ link cannot deliver the message to its target bus destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist.

You can configure an exception destination for a link as one of the following:

- None. The link does not use an exception destination and undeliverable messages are not rerouted to an exception destination. For a service integration bus link, such messages can block the processing of other messages waiting for delivery to the same destination. For a WebSphere MQ link, such messages can block the processing of other messages waiting for delivery through that link to the same bus.
- System. The link uses the default exception destination. Messages that cannot be delivered to the bus destination are rerouted to the system default exception destination for the messaging engine that this link is assigned to: _SYSTEM.Exception.Destination.*messaging_engine_name*. This is the default option.
- Specify. The link uses the specified exception destination. If the link cannot use this exception destination, it uses the system exception destination.

Note that best-effort messages are always discarded if they cannot be delivered to their target destination, that is, they never use an exception destination.

Any report options that are set in the properties of a message also affect exception destination processing. Depending on the report options, a message might be discarded if it is not delivered.

Procedure

1. In the navigation pane of the administrative console, click **Service integration -> Buses** to display a list of buses.
2. Select the bus that has the link for which you want to configure exception destination processing.
3. In the **Configuration** tab, under **Topology**, click **Foreign bus connections** to display a list of connections for this bus.
4. Select the name of the foreign bus connection that is associated with the link for which you want to configure exception destination processing.
5. For a link to another service integration bus, under **Related Items**, click **Service integration bus links**. For a link to a WebSphere MQ network, under **Related Items**, click **WebSphere MQ links**.
6. Select the name of the link you require from the list. The details of that link are displayed.

7. In the **Configuration** tab, under **General properties**, in the Exception destination section, use the radio buttons to configure the exception destination processing that this link uses:
   - Select **None** to specify that the link does not use an exception destination.
   - Select **System** to use the default exception destination.
   - Select **Specify** and enter an exception destination to configure the exception destination you require.
8. Save your changes to the master configuration.

Results

You have configured the exception destination processing for undeliverable messages that are handled by the link.

What to do next

You can also configure exception destination processing for a bus destination.

Managing messages in a link transmission queue for a connection between two buses:

You can manage messages in a link transmission queue that is associated with a foreign bus connection from one service integration bus to another service integration bus.

Before you begin

You must know the link transmitter on which messages are to be managed.

About this task

A service integration bus link can have multiple link transmitters. For applications that use point-to-point messaging, there is one link transmitter for each messaging engine in the sending bus. For applications that use publish/subscribe messaging, there is one link transmitter for each topic space destination that is mapped to a topic space in the foreign bus.

Complete this task when you want to view, delete, or move messages that are queued for a link transmitter.

Procedure

1. From the navigation pane of the administrative console, click **Service integration -> Buses** to display a list of buses.
2. Select the bus whose link transmission queue you want to manage.
3. In the **Configuration** tab, under **Topology**, click **Foreign bus connections** to display a list of connections for this bus.
4. From the list of foreign bus connections, select the name of a foreign bus connection to display its details.
5. Under **Related Items**, click **Service integration bus links**, then select the check box for a link.
6. In the **Runtime** tab, click **Link transmitters** to display the list of link transmitters for messaging engines that are sending messages to the foreign bus.
7. Optional: To manage all messages that are queued for a link transmitter, select the link transmitter and complete your required task:
   - Click **Delete all messages** to delete all the available messages that are queued for a selected link transmitter.
   - Click **Move all messages** to move all the available messages that are queued for a selected link transmitter to the configured exception destination. If the exception destination is configured as

None, or if the link transmitter is a publish/subscribe link transmitter, the messages are discarded. If it is configured as **System**, the messages are sent to the system default exception destination. If a specified destination is configured, the messages are sent to the exception destination specified for this link.

8. Optional: To manage selected messages that are queued for a link transmitter, click the name of the messaging engine from the list of link transmitters, and then click **View messages** to view the messages on an outbound stream from this link transmitter to the foreign bus.
9. Select one or more messages and complete your required task:
   - Click **Delete** to delete the selected messages.
   - Click **Move** to move the selected messages to the local exception destination. If the exception destination is configured as **None**, or if the link transmitter is a publish/subscribe link transmitter, the messages are discarded.
   - Click **Refresh** to view the current messages that are queued for a selected link transmitter.

     Note: Only messages up to the specified maximum displayed messages are displayed.

Results

You have managed the messages that are queued for a specified link transmitter for a specified foreign bus connection.

Viewing messages in a link receiver queue for a connection between two buses:

You can view messages in a link receiver queue that is associated with a foreign bus connection from one service integration bus to another service integration bus.

Before you begin

You must know the link receiver on which messages are to be viewed.

About this task

A service integration bus link can have multiple link receivers. For applications that use point-to-point messaging, there is one link receiver for each messaging engine in the foreign bus, that is, the bus that sends messages through the link. For applications that use publish/subscribe messaging, there is one link receiver for each topic space in the foreign bus.

In this situation, the bus with the link receiver is the local bus. The bus that the applications that produce messages are connected to is the foreign bus.

Complete this task when you want to view messages on a link receiver.

Procedure

1. From the navigation pane of the administrative console, click **Service integration -> Buses** to display a list of buses.
2. Select the bus that has the link receiver queue you want to view.
3. In the **Configuration** tab, under **Topology**, click **Foreign bus connections** to display a list of connections for this bus.
4. From the list of foreign bus connections, select the name of the foreign bus connection to display its details.
5. Under **Related links** click **Service integration bus links**, then select the check box for a link.
6. In the **Runtime** tab, click **Link receivers** to display the list of link receivers for messaging engines on the foreign bus that are sending messages to this bus.

7. To view the messages on the link receiver, select the name of the messaging engine from the list of link receivers, then click **View messages** to view the messages on an inbound stream from the messaging engine on the foreign bus. Click **Refresh** to view the current messages on the selected link receiver.

Results

You have viewed the messages on a specified link receiver queue for a specified foreign bus connection.

Managing messages in a link transmission queue for a connection to a WebSphere MQ network:

You can manage messages in a link transmission queue that is associated with a foreign bus connection from a service integration bus to a WebSphere MQ network.

Before you begin

You must know the foreign bus connection and the link transmitter for which you want to manage messages.

About this task

A link can have multiple link transmitters; one for each messaging engine that sends messages on the link to the foreign bus, and one for each topic space destination that is mapped to a topic space in the foreign bus.

Complete this task when you want to view, delete, or move messages that are queued for a link transmitter.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses** to display a list of buses.
3. Select the bus whose link transmission queue you want to manage.
4. In the **Configuration** tab, under **Topology**, click **Foreign bus connections** to display a list of connections for this bus.
5. Select the name of the foreign bus connection to display its details.
6. Under **Related Items**, click **WebSphere MQ links**, then select the name of the link.
7. In the **Runtime** tab, click **Sender channel** to display the sender channels that are sending messages to the foreign bus.
8. Optional: To manage all messages that are queued for a link transmitter, select the link transmitter and complete your required task:
   - Click **Delete all messages** to delete all the available messages that are queued for the selected link transmitter.
   - Click **Move all messages** to move all the available messages that are queued for the selected link transmitter to the configured exception destination. If the exception destination is configured as **None**, or if the link transmitter is a publish/subscribe link transmitter, the messages are discarded. If it is configured as **System**, the messages are sent to the system default exception destination. If a specified destination is configured, the messages are sent to the exception destination specified for this link.
9. Optional: To manage selected messages that are queued for a link transmitter, click the name of the messaging engine from the list of link transmitters and then click **View messages** to view the messages on an outbound stream from this link transmitter to the foreign bus.
10. Select one or more messages and complete one of the following tasks:
    - Click **Delete** to delete the selected messages.

- Click **Move** to move the selected messages to the local exception destination.If the exception destination is configured as **None**, or if the link transmitter is a publish/subscribe link transmitter, the messages are discarded.
- Click **Refresh** to view the current messages that are queued for the selected link transmitter.

  Note: Only messages up to the specified **Maximum displayed messages** are displayed.

Results

You have managed the messages that are queued for a specified link transmitter for a specified foreign bus connectoin.

Managing messages in a known link transmission queue for connection to a WebSphere MQ network:

You can manage messages in a known link transmission queue that is associated with a foreign bus connection from a service integration bus to a WebSphere MQ network.

Before you begin

You must know the foreign bus connection and the known link transmitter for which for which you want to manage messages.

About this task

A link can have multiple known link transmitters; one for each messaging engine in the bus that has sent messages through the link to the foreign bus, and one for each topic space destination that is mapped to a topic space in the foreign bus.

Complete this task when you want to view, delete, or move messages that are queued for a known link transmitter.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses** to display a list of buses.
3. Select the bus whose known link transmission queue you want to manage.
4. In the **Configuration** tab, under **Topology**, click **Foreign bus connections** to display a list of connections for this bus.
5. Select the name of the foreign bus connection to display its details.
6. Under **Related Items** click **WebSphere MQ links**, then select the name of the link.
7. In the **Configuration** tab, under **Additional Properties**, click **Sender channel** to display the sender channels for all messaging engines that are sending messages to this bus. The sender channel sends messages to the gateway queue manager.
8. From the list of sender channels, select the check box of a **Sender MQ channel name** and click **View known link transmitters** to display a list the remote messaging engines producing messages for this WebSphere MQ link.
9. Click **Refresh** to view the current messages on the message engines.
10. Optional: To manage all messages that are queued for a known link transmitter, select a known link transmitter and complete your required task:
    - Click **Delete all messages** to remove the messages from the known link transmission queue.
    - Click **Move all messages** to move all the available messages that are queued for a selected known link transmitter to the configured exception destination. If the exception destination is configured as **None**, or if the known link transmitter is a publish/subscribe known link transmitter, the messages are discarded. If it is configured as **System default** the messages are sent to the

system default exception destination. If a **Specified destination** is configured, the messages are sent to the exception destination specified for this link.

11. Optional: To manage selected messages that are queued for a known link transmitter, select a known link transmitter then click **View messages** to view the messages.
12. Select one or more messages and complete your required task:
    - Click **Delete** to remove the selected messages from the known link transmission queue.
    - Click **Move** to move the selected messages to the configured exception destination. If the exception destination is configured as **None**, or if the known link transmitter is a publish/subscribe known link transmitter, the messages are discarded. If the exception destination is configured as **System default**, the messages are sent to the system default exception destination. If a **Specified destination** is configured, the messages are sent to the exception destination specified for this link.

Results

You have managed the messages in a known link transmission queue for a specified foreign bus connection between a service integration bus and a WebSphere MQ network.

Managing messages in a sender channel transmission queue for a connection to a WebSphere MQ network:

You can manage messages in a sender channel transmission queue that is associated with a foreign bus connection between a service integration bus and a WebSphere MQ network.

Before you begin

You must know the foreign bus connection and the sender channel transmitter for which you want to manage messages.

About this task

There is one sender channel transmitter on the WebSphere MQ link that sends messages to a gateway queue manager on the WebSphere MQ network (the foreign bus).

Complete this task when you want to view, delete, or move messages that are queued for the sender channel transmitter.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses** to display a list of buses.
3. Select the bus whose sender channel transmission queue you want to manage.
4. In the **Configuration** tab, under **Topology**, click **Foreign bus connections** to display a list of connections for this bus.
5. Select the name of the foreign bus connection to display its details.
6. In the **Configuration** tab, under **Related Items** click **WebSphere MQ links**, then select the check box for a link.
7. Click **Sender channel transmitter** under **Current outbound messages**, or click **Sender channel transmitter** under **Messages sent**, to see the relevant message details for this sender channel transmitter.
8. Select the check box of the sender channel transmitter and complete your required task:
    - Click **Delete all messages** to remove the messages from the sender channel transmission queue.
    - Click **Move all messages** to move all the available messages on the sender channel transmitter to the configured exception destination. If the exception destination is configured as **None**, the

messages are discarded. If it is configured as **System**, the messages are sent to the system default exception destination. If a specified destination is configured, the messages are sent to the exception destination specified for this link.

Results

You have managed the messages on a sender channel transmitter that sends messages to a gateway queue manager on the WebSphere MQ network.

Managing pending acknowledgement messages on a deleted WebSphere MQ link:

If a foreign bus link to a WebSphere MQ network is deleted from the WebSphere Application Server configuration before being drained of messages, a batch of messages pending acknowledgement remains stored in the WebSphere MQ link sender channel transmitter. You can use the administrative console to resolve these messages.

Before you begin

You must know the name of the WebSphere MQ link that has been deleted.

About this task

To resolve pending acknowledgement messages on a deleted WebSphere MQ link, use the administrative console to complete the following steps.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses** to display a list of buses.
3. Select the bus whose link transmission queue you want to manage.
4. In the **Configuration** tab, under **Topology**, click **Foreign Bus Connections** to display a list of connections for this bus.
5. From the list of foreign bus connections, select the name of a foreign bus connection to display its details.
6. Under **Related Items** click **Service integration bus links** to display the details of the service integration bus links.
7. Select the MQ network foreign bus that has a connection that is active, but a configuration status that is **Deleted**. If clicking **Link transmitters** displays an empty list, no messaging engines are producing messages to this link and all the link transmitters have been deleted because they were drained of messages. The **Sender channel transmitter** link displays the status of the sender channel as stopped but **Current outbound messages** shows remaining messages on the sender channel transmitter.
8. Click a WebSphere MQ link **Sender channel** link to display the messages that are queued on the WebSphere MQ link sender channel transmitter for transmission to the WebSphere MQ network.
9. If the Status of a batch of messages is "Commit pending batch", the batch has arrived safely at the MQ network. Select the batch and click **Commit pending batch** to remove the messages from the transmission queue.
10. If the Status of a batch of messages is "Pending batch acknowledgement", the batch did not arrive at the MQ network. Select the batch and click **Rollback pending acknowledge batch** to roll back the transaction and restore the messages to the channel in an available state. These messages are either automatically deleted or moved to the exception destination. When the channel transmitter is empty, the link is automatically deleted from the runtime environment.

Results

You have resolved any pending acknowledgement messages on a WebSphere MQ link that has been deleted from a foreign bus connecting to a service integration bus.

Modifying a routing definition:

You can view or change the routing definition of an existing foreign bus connection between a local bus and a foreign bus. The routing definition defines the virtual link between two buses that enables them to exchange messages. The routing definition can define properties for a virtual service integration bus link, a virtual WebSphere MQ link, or an indirect foreign bus connection.

About this task

You create a routing definition when you create a foreign bus connection. For a direct foreign bus connection, the routing definition, or virtual link, has a corresponding physical link on a messaging engine.

To change the properties of the routing definition, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that is connected to another bus, that is, the local bus you require.
3. In the content pane, under **Topology**, click **Foreign bus connections**. A list of buses that have a foreign bus connection from the local bus is displayed.
4. Select the required foreign bus.
5. In the content pane, under **Additional properties**, click the routing properties option. The routing properties option depends on the type of routing definition and is one of the following:
   - **Service integration bus link routing properties**
   - **WebSphere MQ link routing properties**
   - **Indirect routing properties**
6. Specify the new routing properties:
   - For a virtual service integration bus link or a virtual WebSphere MQ link, specify properties as follows:

     Inbound user ID
     The user name used to authenticate inbound message flows from the foreign bus.

     When the local bus is secure, the inbound user ID replaces the user ID in messages from the foreign bus that arrive at the local bus and is used to authorize whether those messages can access their destinations. Specify an inbound user ID for the local service integration bus under the following circumstances:

     – The foreign bus is in a different security domain, so user IDs in the foreign bus are not recognized in the local bus.
     – You want local control of access to inbound messages to the local bus.

     If the local bus is not secure, the inbound user ID has no effect on messages. If the local bus is secure, the foreign bus is not secure, and an inbound user ID is not set, an inbound message from the foreign bus is only authorized to destinations that allow unauthenticated users access.

     Outbound user ID
     The user name used to authenticate outbound message flows to the foreign bus.

The outbound user ID replaces the user ID that identifies the message source in every message sent to the foreign bus. Where it is defined, the outbound user ID replaces the user ID in messages sent by the local bus to the foreign bus. If the local bus and the foreign bus are both secure, and the foreign bus has not overridden the user ID with its own inbound user ID, the foreign bus also uses the outbound user ID to authorize the message to its destination.

- For an indirect foreign bus connection, select the name of the next foreign bus you require in the chain of intermediate buses. For a bus to be available for selection, it must already have a direct foreign bus connection from the local bus.

7. Click **OK**.
8. Save your changes to the master configuration.

Configuring service integration bus links:

You can configure service integration bus links on messaging engines in a variety of ways. For example, you can start, stop, or remove links.

About this task

When you create a foreign bus connection to connect two service integration buses, a service integration bus link is created automatically. The foreign bus connection contains a routing definition, which is a virtual link, and the service integration bus link is the corresponding physical link on the messaging engine.

Configuring the properties of a service integration bus link:

After establishing a service integration bus link you might want to configure the properties of a service integration bus link such as the name of the service integration bus link, or authentication alias used by foreign bus that the link connects to.

About this task

To configure the properties of a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine for which you want to configure the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to configure.
5. Specify the following properties for the service integration bus link:

   Name The name of the service integration bus link. In order to work, the name must be the same as the name of the corresponding service integration bus link configured on the target foreign bus.

   Description
   An optional description for the service integration bus link, for administrative purposes.

   UUID The universal unique identifier assigned by the system to the service integration bus link for administrative purposes.

   Foreign messaging engine
   The messaging engine on the foreign bus to which this link connects.

Note: This foreign bus name must not be altered after it has been configured. If you alter it, any messaging engines that already hold state information on the link will not be able to use the link unless the foreign bus name is reset to its original value.

Target inbound transport chain
The type of transport chain used for communication with the foreign bus. The transport chain name must be the name of the transport chain as defined on the server on which the target messaging engine is hosted.

Bootstrap endpoints
The comma-separated list of endpoints used to connect to a bootstrap server. This property is set in the same way as the **Provider endpoint** property in the JMS connection factory settings. For more information, see the steps relating to setting bootstrap endpoints in "Configuring a connection to a non-default bootstrap server" on page 531.

Note: Service integration bus links over BootstrapTunneledMessaging and BootstrapTunneledSecureMessaging transport chains only work directly between application server instances. Bus links over TunneledMessaging transport chains do not work if an HTTP server is placed in front of either application server instance.

Authentication alias
The name of the authentication alias, used to authenticate access to the foreign bus. The alias must be known to the foreign bus.

Initial state
Whether the link is started automatically when the messaging engine is started. Until started, the gateway link is unavailable. If this property is set to **Started** the service integration bus link is started when the messaging engine is started.

6. Click **OK**.
7. Save your changes to the master configuration.

Listing the service integration bus links:

You can list all the service integration bus links that are linked to a messaging engine.

About this task

To list the service integration bus links for a messaging engine, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that your messaging engine belongs to.
3. In the content pane, under **Additional properties**, click **Messaging engines**. The list of messaging engines in the bus is displayed.
4. In the content pane, select the messaging engine for which you want to list the service integration bus links.
5. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.

Example

The following combinations of **Status** and **Activity** values are possible:

Table 206. Results of possible status and activity values. The first and second columns of the table provide the possible combinations of status and activity values, and the third column explains the possible connection status of the local and foreign buses.

Status	Activity	Meaning
started	inactive	The service integration bus link is started on the local messaging engine but has no connection to the foreign bus. The service integration bus link is attempting to activate a connection to the foreign bus. The service integration bus link on the foreign bus must also be started to successfully activate of a connection between the buses.
started	active	The service integration bus link is started on the local messaging engine and has an active connection to the foreign bus.
stopped	inactive	The service integration bus link is stopped on the local messaging engine and there is no connection to the foreign bus.
unknown	inactive	An error might have occurred in setting up the link, such that the object that is used to report the current state is not available.

What to do next

You can now add or remove a service integration bus link or select a service integration bus link to start, stop, or configure.

Starting a service integration bus link:

When a service integration bus link has been started, it can be used for communicating with its associated messaging engines.

About this task

To start a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine for which you want to start the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to start.
5. Click **Start**.

Stopping a service integration bus link:

When a service integration bus link has been stopped, it cannot be used for communication until it is restarted.

About this task

To stop a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine to which the service integration bus link that you want to stop belongs.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to stop.
5. Click **Stop**.

Removing a service integration bus link:

When a service integration bus link to a messaging engine has been removed, it cannot be used for communication until it is recreated.

Before you begin

Before you remove the service integration bus link you must stop the link.

When you remove a service integration bus link, all traffic that uses the link needs to be dealt with in a similar way to when you remove a destination. For further details, see “Deleting a non-topic space bus destination” on page 1997. Note that express Quality of Service (QoS) messages are discarded.

About this task

To remove a service integration bus link from a messaging engine, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine that is associated with the service integration bus link that you want to remove.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to remove.
5. Click **Delete**.
6. Save your changes to the master configuration.

What to do next

When you remove a service integration bus link, all traffic that uses the link must be diverted by alternative routes or held pending further administrative action. Express QoS messages are discarded.

Configuring topic space mappings between service integration buses:

A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. You can configure topic space mappings between a local service integration bus and a foreign service integration bus.

About this task

A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. For publications to flow from the local topic space into the foreign bus, an equivalent topic space mapping is required by the foreign bus.

If you want to publish messages from a local bus to a foreign bus that is linked indirectly through a third bus, you must configure topic space mappings between the local bus and the intermediate bus, and between the intermediate bus and the foreign bus. The messages can be for publication to the foreign bus only, or to all three buses. If you want to publish messages only to the local and foreign buses, these two buses must be connected directly.

If you want to publish messages from a local service integration bus to a foreign bus that represents a WebSphere MQ network, you must define topic mappings on the broker profile associated with the WebSphere MQ link.

For more details about the foreign bus and publish/subscribe messaging between buses, see Foreign buses.

Creating topic space mappings:

A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. You can create topic space mappings between two service integration buses.

About this task

You can create topic space mappings as part of the procedure to create a foreign bus connection, or you can create them separately, as described in this topic.

Note that for publishing and subscribing between a service integration bus and a foreign bus that represents a WebSphere MQ network, you must define topic mappings on the broker profile associated with the WebSphere MQ link.

To map a topic space on a local service integration bus to a topic space on a foreign service integration bus, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that is connected to another bus, that is, the local bus you require.
3. In the content pane, under **Topology**, click **Foreign bus connections**. A list of buses that have a foreign bus connection from the local bus is displayed.
4. Select the required foreign bus.
5. In the content pane, under **Additional properties**, click **Service integration bus link routing properties**.
6. In the content pane, under **Additional properties**, click **Topic space map entries**.
7. Click **New**.
8. Specify the mappings as follows:

   Local topic space
   The name of the topic space on this (local) bus that is mapped to the remote topic space on the foreign bus.

   Remote topic space
   The name of the topic space on the foreign bus that is mapped to the local topic space.

   Note: The names of the local and foreign topic space do not have to match, but the names of the topics must match in both local and foreign buses.
9. Click **OK**.
10. Save your changes to the master configuration.

Results

The remote topic space is mapped to the local topic space.

Deleting topic space mappings:

You can delete topic space mappings so that subscribers on the local topic space cannot receive messages that are published in the foreign topic space.

About this task

To delete topic space mappings, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that is connected to another bus, that is, the local bus you require.
3. In the content pane, under **Topology**, click **Foreign bus connections**. A list of buses that have a foreign bus connection from the local bus is displayed.
4. Select the required foreign bus.
5. In the content pane, under **Additional properties**, click **Service integration bus link routing properties**.
6. In the content pane, under **Additional properties**, click **Topic space map entries**. A list of topic space map entries is displayed.
7. Select the topic space map entry that you want to remove.
8. Click **Delete**.
9. Save your changes to the master configuration.

Listing topic space map entries:

A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. You can list topic space map entries to view existing map entries.

Before you begin

There must be a service integration bus link between a local service integration bus and a foreign service integration bus.

About this task

To list the topic space map entries, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**. A list of service integration buses is displayed.
2. In the Buses pane, click the service integration bus that is connected to another bus, that is, the local bus you require.
3. In the content pane, under **Topology**, click **Foreign bus connections**. A list of buses that have a foreign bus connection from the local bus is displayed.
4. Select the required foreign bus.
5. In the content pane, under **Additional properties**, click **Service integration bus link routing properties**.

6. In the content pane, under **Additional properties**, click **Topic space map entries**. The list of mappings between topic spaces in the local bus and topic spaces in the foreign bus is shown.

Topic names and use of wildcard characters in topic expressions:

Wildcard characters can be used in topic expressions to retrieve topics provided by the default messaging provider and service integration technologies.

Each subscribe request includes a topic expression that identifies one or more topics that the subscription is to be associated with, and that the request uses to match against incoming messages.

Subscription topic expressions for the default messaging provider and service integration technologies are based on a subset of the XPath location path syntax.

Identifying individual topics

Every topic in a topic space has a *topic name* consisting of one or more name parts, separated by / (forward slash) characters:

```
Topic name = name_part | (name_part '/' topic_name)
```

Using wildcards to identify multiple topics

To select one or more topics in a topic space, you can use a *topic path*, a location path that contains wildcard characters. Topic spaces are evaluated by using a subset of the XPath location path syntax with the <topicspace> element as the initial context node, so that non-wildcarded topic paths look exactly like topic names.

The syntax for topic paths can be summarized as follows:

- A topic path that contains no * (asterisk), // (double forward slash), or . (dot) symbols is asking for an exact match with the topic name specified.
- A * (asterisk) can be used as a wild card and matches one level (regardless of the value of the name part at that level)

  A * can be used anywhere in a topic path expression, but if it isn't at the start it must be preceded by a /, and if it is isn't at the end it must be followed by a /
- // can be used as a wild card and matches 0 or more levels

  A // can be used anywhere in the expression except at the end. To match 0 or more levels at the end of the expression you end the expression with the syntax //. (double-slash dot). To match one or more levels at the end use //* (double-slash asterisk)

  A topic path must not contain more than two consecutive / symbols.

The following table lists some example topic paths showing the XPath syntax and the equivalent WBI Message Broker selectors:

Table 207. XPath syntax and WBI Message Broker selectors. The first column of the table lists some topic path examples. The second column displays the topics selected in the path. The third column provides the equivalent WBI Message Broker selectors.

Topic path	Topics selected	WBI Message Broker equivalent
A/B	Selects the B child of A	A/B
A/*	Selects all children of A	A/+
A//*	Selects all descendents of A	A/#/+
A//.	Selects A and all descendents of A	A/#
//*	Selects everything	# (or #/+)

Table 207. XPath syntax and WBI Message Broker selectors (continued). The first column of the table lists some topic path examples. The second column displays the topics selected in the path. The third column provides the equivalent WBI Message Broker selectors.

A/./B	Equivalent to A/B	A/B
A/*/B	Selects all B grandchildren of A	A/+/B
A//B	Selects all B descendents of A	A/#/B
//A	Selects all A elements at any level	#/A
*	Selects all first level elements	+

Interoperation with Version 5.1 clients

Existing WebSphere Application Server Version 5.1 client applications using Version 5.1 connection factory and destination definitions use the WBI Message Broker wildcard convention. Such applications can connect to the default messaging provider and service integration bus, and automatically have their wildcard syntax mapped to the XPath convention when subscriptions are created. Any display of these subscriptions through an administrative interface to the bus shows the XPath syntax.

Defining outbound chains for bootstrapping

You can define new outbound chains by using the wsadmin utility. These chains can be used for bootstrapping connections to messaging engines.

About this task

To create outbound chains for bootstrapping, there are several main steps:

1. Locate the appropriate TransportChannelService configuration object. This object is the parent object of all the objects created.
2. Create the individual channels that comprise the transport channel service. Some of these channels might require references to other configuration objects.
3. Assemble the channels that you have created into an outbound channel chain.

The channels used to build an outbound bootstrap chain determine the protocol with which the outbound chain can be used to bootstrap. The following table shows all valid bootstrap chains with their bootstrap protocols.

Table 208. Valid bootstrap chains and protocols. The first column contains the bootstrap protocol used in building the outbound bootstrap chain. The second to fifth columns in the table contain the channels that are valid for the protocol specified on the first column. The order of the channels is important while building the chain. The order of the channels from left to right as given in the table is TCP, SSL, HTTP, HTTP tunneling, and JFAP channels.

Bootstrap protocol	TCP channel	SSL channel	HTTP channel	HTTP tunneling channel	JFAP channel
TCP	X				X
SSL	X	X			X
HTTP	X		X	X	X
HTTPS	X	X	X	X	X

For example, a chain for bootstrapping that uses the SSL protocol would consist of a TCP channel, SSL channel, and JFAP channel. When you create chains, the order of channels in the chain is important. You must specify channels in the order (left to right) in which they appear in the preceding table.

The example in this topic describes how to create a bootstrap chain that is capable of bootstrapping by using the HTTPS protocol. This requires a chain containing all the channel types described. Thus, it is easy to see how to create chains for other protocols by omitting channels during the chain creation step.

Note: You open a wsadmin command session from within Qshell. For more information, see the topic "Configure Qshell to run WebSphere Application Server scripts".

Procedure

1. Locate the TransportChannelService object for the server in which you want to create the new chain. For example, in a WebSphere Application Server Network Deployment configuration, you can list the available TransportChannelService objects and select the appropriate service as follows.

   Using Jython:

```
wsadmin>AdminConfig.list("TransportChannelService" )
(cells/BadgerCell01/nodes/BadgerCellManager01/servers/dmgr|server.xml
#TransportChannelService_1)
(cells/BadgerCell01/nodes/BadgerNode01/servers/nodeagent|server.xml
#TransportChannelService_1095
711814579)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml
#TransportChannelService_109571
2023139)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server2|server.xml
#TransportChannelService_109571
2039302)
wsadmin>tcs = AdminConfig.list("TransportChannelService" ).split("\r\n")[2]
```

   Using Jacl:

```
wsadmin> $AdminConfig list TransportChannelService
(cells/BadgerCell01/nodes/BadgerCellManager01/servers/dmgr|server.xml
#TransportChannelService_1)
(cells/BadgerCell01/nodes/BadgerNode01/servers/nodeagent|server.xml
#TransportChannelService_1095
711814579)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml
#TransportChannelService_109571
2023139)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server2|server.xml
#TransportChannelService_109571
2039302)
wsadmin> set tcs [lindex [$AdminConfig list TransportChannelService] 2]
(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
TransportChannelService_1095712023139)
```

2. Define an outbound TCP channel called testTCPChannel.

   Using Jython:

```
wsadmin>tcpChannel = AdminConfig.create("TCPOutboundChannel", tcs,
[["name", "testTCPChannel"]] )
```

   Using Jacl:

```
wsadmin>set tcpChannel [$AdminConfig create TCPOutboundChannel $tcs
"{name testTCPChannel}"]
testTCPChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/
server1|server.xml#TCPOutboundChannel_1095969213949)
```

3. Define an outbound SSL channel called testSSLChannel. There are two steps required to define such a channel.

   a. Identify the SSL alias to be used by the channel.

      Using Jython:

```
wsadmin>for obj in AdminConfig.list("SSLConfig" ).split("\r\n"):
print obj+AdminConfig.show(obj, "alias")
(cells/BadgerCell01|security.xml#SSLConfig_1)
[alias BadgerCellManager01/DefaultSSLSettings]
(cells/BadgerCell01|security.xml#SSLConfig_1095711819776)
[alias BadgerNode01/DefaultSSLSettings]
```

      Using Jacl:

```
wsadmin>foreach obj [$AdminConfig list SSLConfig] { puts "$obj
[$AdminConfig show $obj alias]]" }
(cells/BadgerCell01|security.xml#SSLConfig_1) {alias BadgerCellManager01/
DefaultSSLSettings}]
(cells/BadgerCell01|security.xml#SSLConfig_1095711819776) {alias BadgerNode01/
DefaultSSLSettings}]
```

b. Create an SSL channel as in the following example, in which the BadgerNode01/DefaultSSLSettings alias is used.

Using Jython:

```
wsadmin>sslChannel =
AdminConfig.create("SSLOutboundChannel", tcs, [["name", "testSSLChannel"],
["sslConfigAlias", "BadgerNode01/DefaultSSLSettings"]] )
```

Using Jacl:

```
wsadmin>set sslChannel [$AdminConfig create SSLOutboundChannel $tcs
"{name testSSLChannel}
{sslConfigAlias BadgerNode01/DefaultSSLSettings}"]
testSSLChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
SSLOutboundChannel_1095971760671)
```

4. Define an outbound HTTP channel called testHTTPChannel.

Using Jython:

```
wsadmin>httpChannel = AdminConfig.create("HTTPOutboundChannel", tcs,
[["name", "testHTTPChannel"]] )
```

Using Jacl:

```
wsadmin>set httpChannel [$AdminConfig create HTTPOutboundChannel $tcs
"{name testHTTPChannel}"]
testHTTPChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
HTTPOutboundChannel_1095971896556)
```

5. Define an outbound HTTP tunneling channel called testHTCChannel.

Using Jython:

```
wsadmin>htcChannel = AdminConfig.create("HTTPTunnelOutboundChannel", tcs,
[["name", "testHTCChannel"]] )
```

Using Jacl:

```
wsadmin>set htcChannel [$AdminConfig create HTTPTunnelOutboundChannel $tcs
"{name testHTCChannel}"]
testHTCChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
HTTPTunnelOutboundChannel_1095972164201)
```

6. Define an outbound JFAP channel called testJFAPChannel.

Using Jython:

```
wsadmin>jfapChannel = AdminConfig.create("JFAPOutboundChannel", tcs,
[["name", "testJFAPChannel"]] )
```

Using Jacl:

```
wsadmin>set jfapChannel [$AdminConfig create JFAPOutboundChannel $tcs
"{name testJFAPChannel}"]
testJFAPChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
JFAPOutboundChannel_1095972226631)
```

7. Finally, create the channel chain by combining the channels defined so far. For example, to create a chain called testChain:

Using Jython:

```
wsadmin>AdminConfig.create("Chain", tcs, [["name", "testChain"], ["enable", "true"],
["transportChannels", [tcpChannel, httpChannel, htcChannel, jfapChannel]]] )
testChain(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
Chain_1095972662147)
```

Using Jacl:

```
wsadmin>$AdminConfig create Chain $tcs "{name testChain} {enable true}
{transportChannels {$tcpChannel $httpChannel $htcChannel $jfapChannel}}"
testChain(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
Chain_1095972662147)
```

Defining outbound chains for WebSphere MQ interoperation

You can define new outbound chains by using the wsadmin utility. These chains can be used for interoperating with WebSphere MQ.

About this task

The channels used to build an outbound chain determine with which configurations of the WebSphere MQ queue manager sender channel so a network connection can be successfully established. The following table shows all the valid chain configurations and describes the configuration of aWebSphere MQ queue manager sender channel with which they can be used to establish a connection.

Table 209. Valid chain and WebSphere MQ queue manager sender channel configurations. The first column of the table provides the WebSphere MQ channels. The second to fourth columns in the table indicate whether the TCP, SSL, and MQFAP channels in combination with the WebSphere MQ channel can establish a network connection successfully. It is important to follow a specific order of the channels while building the chain. The order of the channels from left to right as given in the table is TCP, SSL, and MQFAP channels.

WebSphere MQ channel	TCP channel	SSL channel	MQFAP channel
Unsecured WebSphere MQ channel	X		X
WebSphere MQ channel secured by using SSL	X	X	X

For example, an SSL-based chain would consist of a TCP channel, SSL channel and MQFAP channel. When creating chains, the order of channels in the chain is important. You must specify channels in the order (left to right) in which they appear in the above table.

The example in this topic describes how to create an outbound chain capable of being used to contact WebSphere MQ queue manager receiver channels by using SSL-based encryption.

Note: You open a wsadmin command session from within Qshell. For more information, see the topic "Configure Qshell to run WebSphere Application Server scripts".

Procedure

1. Locate the TransportChannelService object for the server in which you want to create the new chain. For example, in a WebSphere Application Server Network Deployment configuration, you can list the available TransportChannelService objects and select the appropriate service.

   Using Jython:

```
wsadmin>AdminConfig.list("TransportChannelService" )
(cells/BadgerCell01/nodes/BadgerCellManager01/servers/dmgr|server.xml
#TransportChannelService_1)
(cells/BadgerCell01/nodes/BadgerNode01/servers/nodeagent|server.xml
#TransportChannelService_1095
711814579)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml
#TransportChannelService_109571
2023139)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server2|server.xml
#TransportChannelService_109571
2039302)
wsadmin>tcs = AdminConfig.list("TransportChannelService" ).split("\r\n")[2]
```

   Using Jacl:

```
wsadmin> $AdminConfig list TransportChannelService
(cells/BadgerCell01/nodes/BadgerCellManager01/servers/dmgr|server.xml
#TransportChannelService_1)
(cells/BadgerCell01/nodes/BadgerNode01/servers/nodeagent|server.xml
#TransportChannelService_1095711
814579)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml
#TransportChannelService_109571202
3139)
(cells/BadgerCell01/nodes/BadgerNode01/servers/server2|server.xml
#TransportChannelService_109571203
9302)
```

```
wsadmin> set tcs [lindex [$AdminConfig list TransportChannelService] 2]
(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml
#TransportChannelService_109571202
3139)
```

2. Define an outbound TCP channel called testTCPChannel.

   Using Jython:

```
wsadmin>tcpChannel = AdminConfig.create("TCPOutboundChannel", tcs,
[["name", "testTCPChannel"]] )
```

   Using Jacl:

```
wsadmin>set tcpChannel [$AdminConfig create TCPOutboundChannel $tcs
"{name testTCPChannel}"]
testTCPChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
TCPOutboundChannel_1095969213949)
```

3. Define an outbound SSL channel called testSSLChannel. There are two steps required to define such a channel.

   a. Identify the SSL alias to be used by the channel.

      Using Jython:

```
wsadmin>for obj in AdminConfig.list("SSLConfig" ).split("\r\n"):
print obj+AdminConfig.show(obj, "alias")
```

      Using Jacl:

```
wsadmin>foreach obj [$AdminConfig list SSLConfig] { puts "$obj
[$AdminConfig show $obj alias]]" }
(cells/BadgerCell01|security.xml#SSLConfig_1) {alias BadgerCellManager01/
DefaultSSLSettings}]
(cells/BadgerCell01|security.xml#SSLConfig_1095711819776) {alias BadgerNode01/
DefaultSSLSettings}]
```

   b. Create an SSL channel as in the following example, in which the BadgerNode01/DefaultSSLSettings alias is used.

      Using Jython:

```
wsadmin>sslChannel = AdminConfig.create("SSLOutboundChannel", tcs, [["name",
"testSSLChannel"], ["sslConfigAlias","BadgerNode01/DefaultSSLSettings"]])
```

      Using Jacl:

```
wsadmin>set sslChannel [$AdminConfig create SSLOutboundChannel $tcs
"{name testSSLChannel}
{sslConfigAlias BadgerNode01/DefaultSSLSettings}"]
testSSLChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
SSLOutboundChannel_1095971760671)
```

4. Define an outbound MQFAP channel called testMQFAPChannel.

   Using Jython:

```
wsadmin>mqfapChannel = AdminConfig.create("MQFAPOutboundChannel", tcs,
[["name", "testMQFAPChannel"]] )
```

   Using Jacl:

```
wsadmin>set mqfapChannel [$AdminConfig create MQFAPOutboundChannel $tcs
"{name testMQFAPChannel}"]
testMQFAPChannel(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#
MQFAPOutboundChannel_1095977512682)
```

5. Finally, create the channel chain by combining the channels defined so far. For example, to create a chain called testChain:

   Using Jython:

```
wsadmin>AdminConfig.create("Chain", tcs, [["name", "testChain"], ["enable",
"true"], ["transportChannels", [tcpChannel, sslChannel, mqfapChannel]]] )
```

   Using Jacl:

```
wsadmin>$AdminConfig create Chain $tcs "{name testChain} {enable true}
{transportChannels {$tcpChannel $sslChannel $mqfapChannel}}"
testChain(cells/BadgerCell01/nodes/BadgerNode01/servers/server1|server.xml#Chain_109
5977640896)
```

Operating buses

Use these tasks to operate service integration buses.

About this task

The topics in this section describe how to display the runtime properties of messaging engines and their associated service integration bus links. The properties are influenced by the following operations:

- Starting and stopping messaging engines.
- Starting and stopping service integration bus links.

Displaying the runtime properties of a messaging engine

Display the runtime properties of a messaging engine by using the administrative console.

Before you begin

To be able to retrieve the status of messaging engines, you must be logged into the administrative console with at least monitor authority. If you do not have this authority, the messaging engine status is displayed as "Unavailable", even if the messaging engine has started. Also, if you are not logged in with the authority needed to retrieve the status of messaging engines, a SECJ0305I error message is logged in the server systemOut log file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

About this task

To display the runtime properties of a messaging engine, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus to which your messaging engine belongs.
3. In the content pane, under **Topology**, click **Messaging engines**. A list of messaging engines is displayed.
4. Click the messaging engine name.
5. Click the **Runtime** tab. The status of the messaging engine, that is, whether it is currently started or stopped is displayed.

Displaying the runtime properties of a service integration bus link

Display the runtime properties of a service integration bus link by using the administrative console.

About this task

To display the runtime properties of a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine that contains the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.

4. Select the service integration bus link whose properties you want to display.
5. Click the **Runtime** tab. The following properties are displayed:

   Status
   The runtime status of the service integration bus link.

   Activity
   Whether the service integration bus link is currently inactive, active, or its activity is unknown.

Managing messages on message points

Use these tasks to list and act on runtime messages that exist on message points in a service integration bus.

About this task

You can list the message points for bus destinations and messaging engines, and list the messages on a selected message point. You can use the list of messages as part of a troubleshooting task to find messages that need to be deleted.

- "Listing messages on a message point" on page 595
- "Deleting messages on a message point" on page 596

Managing service integration buses with administrative commands

You can use these commands to manage service integration buses.

About this task

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and Jython.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Procedure

1. Open a wsadmin command session in local mode For example:

   ```
   wsadmin -conntype none -lang jython
   ```

2. Type `AdminTask.`*`command`*, where *command* is the command format as indicated in the related reference topics.

   For example:

```
wsadmin>AdminTask.listSIBusMembers('[-bus bus1 ]')
[cells/cell01/buses/bus1|sib-bus.xml#SIBusMember_1092155259869]
[cells/cell01/buses/bus1|sib-bus.xml#SIBusMember_1092159844593]
[cells/cell01/buses/bus1|sib-bus.xml#SIBusMember_1092160253751]

wsadmin>AdminTask.listSIBEngines('[-bus bus1 ]')
'node01.server1-bus1(cells/cell01/nodes/node01/servers/server1|sib-engines.xml#
SIBMessagingEngine_1212163145962)\r\n
node02.server2-bus2(cells/cell01/nodes/node02/servers/server2|sib-engines.xml#
SIBMessagingEngine_1212163146273)'
```

Administering messaging engines

These topics provide information about messaging engines, which provide the processing function on a service integration bus.

Procedure

- ../ae/cjk_learning.dita
- "Configuring messaging engines" on page 1897

- “Managing messaging engines with administrative commands” on page 1953
- “SIBAdminCommands: Messaging engine administrative commands for the AdminTask object” on page 2313
- ../ae/rjk_prob0.dita

Configuring messaging engines

You can configure messaging engines in a variety of ways. For example, you can create and apply security to a messaging engine, then use this engine to send and receive messages. When you add a server cluster to a service integration bus, at least one messaging engine is created automatically. If you also use messaging engine policy assistance, some configuration properties are set automatically.

Configuring messaging engine properties

You can configure the properties of a messaging engine in the administrative console. For example you can select whether the messaging engine is started automatically when its associated application server is started, how many messages it can process, and target groups that the engine can join.

About this task

In most cases, you can configure the properties of a messaging engine without interrupting the processing of messages by the messaging engine.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name***.
3. Configure the messaging engine properties. For information about the properties that you can configure, see the property descriptions in “Messaging engines [Settings]” on page 2124
4. Click **OK**.
5. Save your changes to the master configuration.

Listing the messaging engines in a bus

You can view the list of existing messaging engines in a bus by using the administrative console. You can decide which messaging engines you want to change, for example which buses they are associated with.

About this task

To list the messaging engines in a bus, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that your messaging engine belongs to.
3. In the content pane, under **Topology**, click **Messaging engines**. The list of messaging engines in the bus is displayed.

Removing a messaging engine from a bus

You can remove a messaging engine from a service integration bus if you no longer require it to send and receive messages on the bus.

Before you begin

You should be wary of deleting and recreating messaging engines on bus members for which WS-Notification-administered subscribers have been configured, because in some cases this can leave the remote web service subscription active (and passing notification messages to the local server) even

though there is no longer any record of it. For more information, see the WS-Notification troubleshooting tip Problems can occur when deleting administered subscribers and messaging engines.

Procedure

1. Stop the messaging engine. You can stop either in **Immediate** or **Force** mode, as described in “Stopping a messaging engine” on page 1951.
2. Use the wsadmin command deleteSIBEngine to delete the messaging engine. All service integration bus links, MQ links, and custom properties that are owned by the engine are deleted.

   Note: When you remove a messaging engine, WebSphere Application Server does not delete the data store tables automatically. You must remove them manually, or delete all the rows in all the tables. If you do this, a new messaging engine might fail to start if it attempts to use an orphaned data store. Refer to the documentation for your chosen relational database management system for information about deleting tables.

   Alternatively, for Apache Derby, you can delete the database directory, which is located in *profile_root*/databases/com.ibm.ws.sib, where *profile_root* is the directory in which profile-specific information is stored. However, do this only if the messaging engine is the sole user of the database.

   For more information, see Data store life cycle.

   Similarly, the file store files are not automatically deleted when you delete the messaging engine. You might want to delete the file store files to reclaim disk space.

Listing the messaging engines defined for a server bus member

You can display a list of messaging engines defined for a server bus member by using the administrative console. You can decide which messaging engines you want to change, for example, which buses they are associated with.

About this task

To display the list of messaging engines, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click either **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines** or **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engines**. A list of messaging engines is displayed in the content pane.
2. Optional: Select one or more messaging engines to work with, for example to change the properties of the messaging engine.

Creating the database, schema and user ID for a messaging engine

Before the data store for a messaging engine can be set up, you must first create the database, the schema and the database user ID that the messaging engine needs to access the data store tables.

Before you begin

Before you start this task, review the information in Configuration planning for a messaging engine to use a data store, and ensure that you have taken any appropriate action.

About this task

To create the database, schema and user ID for a messaging engine, complete the following steps.

Procedure

1. Create the database for the data store.
2. Create users and schemas in the database. Ensure that the user ID has sufficient privilege to allow the messaging engine to access the data store tables. For more information about the privileges that are required for the selected database, see “Database privileges” on page 1967.
3. If required, create the data store tables by using the data definition language (DDL) statements generated by using the sibDDLGenerator command.

Configuring a messaging engine data store to use a data source:

After configuring a JDBC data source, you can configure a messaging engine data store to use the data source.

Before you begin

To complete this task, you must have chosen or created a bus and a messaging engine, and the messaging engine must specify data store as its message store type.

You must also have configured a data source, as described in “Creating the database, schema and user ID for a messaging engine” on page 1899.

About this task

A messaging engine uses an instance of a JDBC data source to interact with the database that contains the data store for that messaging engine.

Use the WebSphere Application Server administrative console to set the data store configuration parameters.

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name***.
2. Check that the **Message store type** is *Data store*.
3. Click **[Additional Properties] Message store**. The data store configuration detail panel is displayed.
4. Specify the following data store configuration details:

   Data source JNDI name
   Type the JNDI name of the data source that provides access to database that holds the data store.

   Schema name
   Type the name of the database schema that contains the tables used by the data store.

   General tip: The schema name is usually the same as the user ID that is declared in the authentication alias. With some databases, for example DB2, you can provide an alternative schema name. For more information about the relationship between users and schema, refer to the documentation for your chosen RDBMS.

   Informix tip: When you configure your messaging engine to use an Informix database, you must specify the schema name in lowercase letters.

   When it is starting, a messaging engine that uses a data store checks to see if its data store exists. If the **Create tables** option is selected for the configuration, the messaging engine creates the tables in its chosen schema.

   The **Schema name** field is optional. If you require a schema name, consider the following:

- The default schema name is `IBMWSSIB`.
- If you delete the text so that field is blank, the messaging engine takes the user id defined in the authentication alias to be the schema name.
- If you define a schema name explicitly, that schema name is used by the messaging engine.
- If there are multiple messaging engines, you must configure each messaging engine to use a unique schema, otherwise FFDC error messages stating that `Connection cannot be provided as Datasource has been disabled!` might appear. This applies to DB2 in particular.

Authentication alias
Select the authentication alias that enables access to the data source.

Apache Derby Tip: When you create a new Network Attached Apache Derby data store, by default you get a blank authentication alias.

Create tables
Select the check box if you want WebSphere Application Server to create the database tables automatically.

Note: The user ID that the messaging engine uses to connect to the data source must have sufficient authority to create the database tables and indexes.

DB2 for z/OS restriction: Do not select **Create tables** if you are using DB2 for z/OS, otherwise an exception will be thrown when WebSphere Application Server attempts to create the tables.

Number of tables for permanent objects
Permanent tables contain persistent objects for the data store.

Note: You can only increase the number of permanent tables, not decrease them.

Number of tables for temporary objects
Temporary tables contain nonpersistent objects that have been saved to the data store to reduce the messaging engine memory requirement.

Note: You can only increase the number of temporary tables, not decrease them.

Configuring service integration bus links

You can configure service integration bus links on messaging engines in a variety of ways. For example, you can start, stop, or remove links.

About this task

When you create a foreign bus connection to connect two service integration buses, a service integration bus link is created automatically. The foreign bus connection contains a routing definition, which is a virtual link, and the service integration bus link is the corresponding physical link on the messaging engine.

Configuring the properties of a service integration bus link:

After establishing a service integration bus link you might want to configure the properties of a service integration bus link such as the name of the service integration bus link, or authentication alias used by foreign bus that the link connects to.

About this task

To configure the properties of a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine for which you want to configure the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to configure.
5. Specify the following properties for the service integration bus link:

   Name The name of the service integration bus link. In order to work, the name must be the same as the name of the corresponding service integration bus link configured on the target foreign bus.

   Description
   An optional description for the service integration bus link, for administrative purposes.

   UUID The universal unique identifier assigned by the system to the service integration bus link for administrative purposes.

   Foreign messaging engine
   The messaging engine on the foreign bus to which this link connects.

   Note: This foreign bus name must not be altered after it has been configured. If you alter it, any messaging engines that already hold state information on the link will not be able to use the link unless the foreign bus name is reset to its original value.

   Target inbound transport chain
   The type of transport chain used for communication with the foreign bus. The transport chain name must be the name of the transport chain as defined on the server on which the target messaging engine is hosted.

   Bootstrap endpoints
   The comma-separated list of endpoints used to connect to a bootstrap server. This property is set in the same way as the **Provider endpoint** property in the JMS connection factory settings. For more information, see the steps relating to setting bootstrap endpoints in "Configuring a connection to a non-default bootstrap server" on page 531.

   Note: Service integration bus links over BootstrapTunneledMessaging and BootstrapTunneledSecureMessaging transport chains only work directly between application server instances. Bus links over TunneledMessaging transport chains do not work if an HTTP server is placed in front of either application server instance.

   Authentication alias
   The name of the authentication alias, used to authenticate access to the foreign bus. The alias must be known to the foreign bus.

   Initial state
   Whether the link is started automatically when the messaging engine is started. Until started, the gateway link is unavailable. If this property is set to **Started** the service integration bus link is started when the messaging engine is started.
6. Click **OK**.
7. Save your changes to the master configuration.

Listing the service integration bus links:

You can list all the service integration bus links that are linked to a messaging engine.

About this task

To list the service integration bus links for a messaging engine, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that your messaging engine belongs to.
3. In the content pane, under **Additional properties**, click **Messaging engines**. The list of messaging engines in the bus is displayed.
4. In the content pane, select the messaging engine for which you want to list the service integration bus links.
5. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.

Example

The following combinations of **Status** and **Activity** values are possible:

Table 210. Results of possible status and activity values. The first and second columns of the table provide the possible combinations of status and activity values, and the third column explains the possible connection status of the local and foreign buses.

Status	Activity	Meaning
started	inactive	The service integration bus link is started on the local messaging engine but has no connection to the foreign bus. The service integration bus link is attempting to activate a connection to the foreign bus. The service integration bus link on the foreign bus must also be started to successfully activate of a connection between the buses.
started	active	The service integration bus link is started on the local messaging engine and has an active connection to the foreign bus.
stopped	inactive	The service integration bus link is stopped on the local messaging engine and there is no connection to the foreign bus.
unknown	inactive	An error might have occurred in setting up the link, such that the object that is used to report the current state is not available.

What to do next

You can now add or remove a service integration bus link or select a service integration bus link to start, stop, or configure.

Starting a service integration bus link:

When a service integration bus link has been started, it can be used for communicating with its associated messaging engines.

About this task

To start a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.

2. In the content pane, select the messaging engine for which you want to start the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to start.
5. Click **Start**.

Stopping a service integration bus link:

When a service integration bus link has been stopped, it cannot be used for communication until it is restarted.

About this task

To stop a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine to which the service integration bus link that you want to stop belongs.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to stop.
5. Click **Stop**.

Removing a service integration bus link:

When a service integration bus link to a messaging engine has been removed, it cannot be used for communication until it is recreated.

Before you begin

Before you remove the service integration bus link you must stop the link.

When you remove a service integration bus link, all traffic that uses the link needs to be dealt with in a similar way to when you remove a destination. For further details, see “Deleting a non-topic space bus destination” on page 1997. Note that express Quality of Service (QoS) messages are discarded.

About this task

To remove a service integration bus link from a messaging engine, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine that is associated with the service integration bus link that you want to remove.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link that you want to remove.
5. Click **Delete**.
6. Save your changes to the master configuration.

What to do next

When you remove a service integration bus link, all traffic that uses the link must be diverted by alternative routes or held pending further administrative action. Express QoS messages are discarded.

Starting a messaging engine

You can either start a messaging engine directly by using the administrative console or by starting the server that hosts the messaging engine.

About this task

To start a messaging engine, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that your messaging engine belongs to.
3. Do one of the following to display a list of messaging engines:
   - For a list of messaging engines in the bus, in the content pane, under **Topology**, click **Messaging engines**.
   - For a list of messaging engines for a server, in the content pane, under **Topology**, click **Bus Members**. Click the name of the required server.
   - For a list of messaging engines for a cluster, in the content pane, under **Topology**, click **Bus Members**. Click the name of the required cluster.
4. Select the messaging engine that you want to start.
5. Click **Start**.

Stopping a messaging engine

You can either stop a messaging engine directly or stop it by stopping the server that hosts the messaging engine.

About this task

A messaging engine can stop in two ways:

Immediate
The messaging engine is stopped on completion of all messaging operations that are being carried out at the time of the stop request.

Force The messaging engine is stopped without allowing messaging operations to complete and applications are forcefully disconnected.

To stop a messaging engine, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that the messaging engine belongs to.
3. Do one of the following to display a list of messaging engines:
   - For a list of messaging engines in the bus, in the content pane, under **Topology**, click **Messaging engines**.
   - For a list of messaging engines for a server, in the content pane, under **Topology**, click **Bus Members**. Click the name of the required server.

- For a list of messaging engines for a cluster, in the content pane, under **Topology**, click **Bus Members**. Click the name of the required cluster.

4. Select the messaging engine that you want to stop.
5. Select **Immediate** or **Force** from the **Stop mode** drop-down list.
6. Click **Stop**.

What to do next

Tip: If an immediate stop is taking too long, you can escalate it to a force stop by selecting the **Force** option. This overrides your previous selection of the **Immediate** option.

Displaying the runtime properties of a messaging engine

Display the runtime properties of a messaging engine by using the administrative console.

Before you begin

To be able to retrieve the status of messaging engines, you must be logged into the administrative console with at least monitor authority. If you do not have this authority, the messaging engine status is displayed as "Unavailable", even if the messaging engine has started. Also, if you are not logged in with the authority needed to retrieve the status of messaging engines, a SECJ0305I error message is logged in the server systemOut log file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

About this task

To display the runtime properties of a messaging engine, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus to which your messaging engine belongs.
3. In the content pane, under **Topology**, click **Messaging engines**. A list of messaging engines is displayed.
4. Click the messaging engine name.
5. Click the **Runtime** tab. The status of the messaging engine, that is, whether it is currently started or stopped is displayed.

Displaying the runtime properties of a service integration bus link

Display the runtime properties of a service integration bus link by using the administrative console.

About this task

To display the runtime properties of a service integration bus link, use the administrative console to complete the following steps:

Procedure

1. Display the list of messaging engines.
2. In the content pane, select the messaging engine that contains the service integration bus link.
3. In the content pane, under **Additional properties**, click **Service integration bus links**. A list of service integration bus links is displayed.
4. Select the service integration bus link whose properties you want to display.
5. Click the **Runtime** tab. The following properties are displayed:

   Status
   The runtime status of the service integration bus link.

   Activity
   Whether the service integration bus link is currently inactive, active, or its activity is unknown.

Managing messaging engines with administrative commands

You can use these commands to manage messaging engines.

About this task

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and Jython.

Procedure

1. Open a wsadmin command session in local mode. For example:

```
wsadmin -conntype none -lang jython
```

   The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

2. Type `AdminTask.`*`command`*, where *command* is the command format as indicated in the related reference topics.

   For example:

```
wsadmin>AdminTask.listSIBusMembers('[-bus bus1 ]')
[cells/cell01/buses/bus1|sib-bus.xml#SIBusMember_1092155259869]
[cells/cell01/buses/bus1|sib-bus.xml#SIBusMember_1092159844593]
[cells/cell01/buses/bus1|sib-bus.xml#SIBusMember_1092160253751]

wsadmin>AdminTask.listSIBEngines('[-bus bus1 ]')
'node01.server1-bus1(cells/cell01/nodes/node01/servers/server1|sib-engines.xml#
SIBMessagingEngine_1212163145962)\r\n
node02.server2-bus2(cells/cell01/nodes/node02/servers/server2|sib-engines.xml#
SIBMessagingEngine_1212163146273)'
```

Administering message stores

A message store enables a messaging engine to preserve operating information and to retain those objects that messaging engines need for recovery in the event of a failure. The default message store for a typical messaging engine is file store. You can also configure a messaging engine to use a data store.

Procedure

- “Administering file stores”
- “Administering data stores” on page 1957

Administering file stores

A file store is a type of message store that directly uses files in the file system through the operating system. Each messaging engine has one and only one file store.

Configuring file store attributes for a messaging engine

When a messaging engine is created, it uses a file store by default. You can configure the file store attributes according to your requirements.

About this task

You set up a messaging engine to use a file store by accepting the default choice of message store when you are creating a bus and adding a new bus member. You can choose either to accept the default settings that the administrative console displays, or to make changes to these settings. After the file store has been created, you can, if required, then make changes to the file store settings.

Modifying file store configuration:

After a file store has been created, you can, if required, modify file store attributes.

About this task

When you add a new bus member that uses a file store, you can choose either to accept the default settings for the file store or make changes to these settings, depending on your requirements.

After the file store has been created, you can, if required, subsequently modify the log size, permanent store size, or temporary store size settings directly through the administrative console. The new values take effect the next time the messaging engine is started.

The directory paths for the file store are fixed when the file store is created, and cannot subsequently be modified. You can create a new file store in a new location. To do this you remove the server or cluster as a bus member and then add it again choosing a new location for the file store. Any messages remaining in the original file store would be lost during this process.

Note: The default configuration for a file store is intended to be sufficient when used in typical messaging workloads. To improve the performance or resilience of the log file, the permanent store file, or the temporary store file, you can modify the file store attributes to control where these files are placed. For example, you can achieve better performance if you place these three file store files on a faster disk. Similarly, you can control the sizes of the log file, the permanent store file, and the temporary store file so that they can handle workloads with a large number of active transactions, large messages, or a large volume of message data resident in the messaging engine.

Procedure

- To make changes to the log size, permanent store size or temporary store size, complete the following steps:
  1. Open the Administrative console.
  2. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Message store**.
  3. Make the changes that you require to the existing log size, permanent store size, or temporary store settings, then click **OK**. For more information about configuring the properties, see “File store [Settings]” on page 2057
  4. Save your changes to the master configuration.
- To choose a new location for a file store, complete the following steps:
  1. To avoid losing messages on destinations that are localized on the file store, first ensure that no messages are left on the queues.
  2. Remove the server or cluster as a member of the bus.
  3. Add the server and cluster as a bus member again, choosing a new location for the file store.

Selecting messaging engine behavior when a file store is full:

You can specify what action a messaging engine takes when a file store is full and applications try to send further messages. You can make application threads wait for the checkpoint to complete, or throw an exception immediately.

About this task

When a file store is full, the messaging engine carries out a checkpoint of the log file to reconcile all message sends and receives since the last checkpoint. This process might take some time to complete. Between the time when the file store becomes full and the time when the checkpoint is complete, if applications try to send a message, the messaging engine throws the exception ObjectStoreFullException and issues message CWSOM1042E.

When an application thread that is sending a message finds that the file store is full, it requests a checkpoint. The default behavior is that the application thread then throws the exception ObjectStoreFullException to the application immediately. You can select an alternative behavior where the application thread does not throw the exception, but waits until the checkpoint has completed. If the checkpoint frees space in the file store, the application thread proceeds and sends the messages before returning. If the file store is still full after the checkpoint, the application thread throws the exception to the application.

Choose to make application threads wait if your applications delete all the messages in the file store, and so they logically know that the file store is no longer full. Although the applications must still wait until the checkpoint is complete, they do not receive exceptions while the checkpoint is being carried out, and they do not have to retry the send.

To change the behavior when the file store is full, use the administrative console to set the value of the property sib.msgstore.storeFullWaitForCheckPoint as follows:

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Custom properties**.
2. Type the name of the property, `sib.msgstore.storeFullWaitForCheckPoint`.
3. Type the value `true` to make application threads wait for the checkpoint to complete before returning. The default value `false` makes application threads throw the exception ObjectStoreFullException immediately after requesting the checkpoint.
4. Click **OK**.
5. Save your changes to the master configuration.

What to do next

Remember: When you change this property, the new value does not take effect until you restart the messaging engine.

Deleting files following removal of a messaging engine:

Removing a messaging engine from the configuration does not automatically delete the file store files. You must also find and delete the log file, the permanent store file, and the temporary store file from the disk to reclaim disk space.

About this task

Files comprising a file store still remain when the messaging engine is deleted. If you are using the default values for the file store directory names, you can delete and then re-create a messaging engine with the same name without manually removing the files. This is due to the presence of the universal unique identifier (UUID) of the messaging engine in the default log and store directory names of a file store.

For example, if the deleted messaging engine is called messagingengine0, and the new one is called messagingengine1, then the log file in the old file store is:

```
C:\{USER_INSTALL_ROOT}\filestores\..\messagingengine0\..\logfile
```

The log file in the new files store is:

```
C:\{USER_INSTALL_ROOT}\filestores\..\messagingengine1\..\logfile
```

The two log files coexist in the same file stores directory. You do not have to delete the old files.

Procedure

1. Locate the directory in which your old file store log files and store files are stored.
2. Delete the redundant files by using the facilities of the operating systems.

Results

After deleting the files left by the redundant file store, free disk space becomes available to your file system.

Backing up and restoring a messaging engine file store

A file store is a type of message store that directly uses files in a file system through the operating system. Therefore, administration of the file store depends on the type of operating system.

About this task

- “Backing up a file store”
- “Restoring a file store”

Backing up a file store:

You can back up the files in a file store by using the facilities of your operating system, or by using a backup tool.

About this task

A file store is a type of message store that directly uses files in a file system through the operating system. Use the facilities of your operating system or a backup tool to back up the log file, the permanent store file, and the temporary store file, which comprise a file store. For more information about these files, see ../ae/cjm1450_.dita.

Important: If you start backing up files while the messaging engine is still running, data might be corrupted.

Note: You must treat the log file, the temporary store file, and the permanent store file as one unit; that is, any operations must be performed on all three files.

Restoring a file store:

You can restore the files in a file store by using the facilities of your operating system, or you can restore the backup copies of your files by using a backup tool.

Before you begin

Before you start this task, make sure that the messaging engine is not running.

Important: If you restore a file store while the messaging engine is still running, data might be corrupted.

About this task

When you complete this task, you must treat the log file, the temporary store file, and the permanent store file (which together comprise a file store) as one unit; that is, any operations must be performed on all three files. For more information about these files, see ../ae/cjm1450_.dita.

Procedure

Use the facilities of your operating system, or a backup tool, to restore the backup copy of the log file, the permanent store file, and the temporary store file.

Administering data stores

A data store is a type of message store that uses a relational database. A data store consists of a set of tables that are in the same database schema. It is used by a messaging engine to store operating information in the database, as well as to preserve essential objects that the messaging engine needs for recovery in the event of a failure.

About this task

A data store consists of the set of tables that a messaging engine uses to store persistent data in a database. See “Data store tables” on page 1967 for a list of the tables that comprise a data store. All the tables in a data store are held in the same database schema. You can create multiple data stores in the same database, provided that you use a different schema name for each data store.

Configuring a messaging engine to use a data store

Although the default message store for a typical messaging engine is file store, you can also configure a messaging engine to use a data store.

About this task

Each messaging engine has its own file store or data store. If the data store is chosen the messaging engine uses an instance of a JDBC data source to interact with the database that contains the data store for that messaging engine.

When a new messaging engine that uses a data store is created on a single server, it is configured to use an Apache Derby data source by default. This enables the messaging engine to run without needing any additional configuration.

If you want to configure a new messaging engine to use your chosen data source when you create that messaging engine on a single server, see “Adding a server as a new bus member” on page 1889.

If you do not want to use the default data source configuration, you can use the WebSphere Application Server administrative console to change the configuration parameters. For example, you can change the data source or you can configure the data store to use a different JDBC provider.

Note: WebSphere Application Server supports direct customer use of the Apache Derby database in *test* environments only. The product does not support direct customer use of Apache Derby database in *production* environments.

Creating the database, schema and user ID for a messaging engine:

Before the data store for a messaging engine can be set up, you must first create the database, the schema and the database user ID that the messaging engine needs to access the data store tables.

Before you begin

Before you start this task, review the information in Configuration planning for a messaging engine to use a data store, and ensure that you have taken any appropriate action.

About this task

To create the database, schema and user ID for a messaging engine, complete the following steps.

Procedure

1. Create the database for the data store.
2. Create users and schemas in the database. Ensure that the user ID has sufficient privilege to allow the messaging engine to access the data store tables. For more information about the privileges that are required for the selected database, see “Database privileges” on page 1967.
3. If required, create the data store tables by using the data definition language (DDL) statements generated by using the sibDDLGenerator command.

Creating the database for a data store:

When you are creating a database to use as the data store for your messaging engine, you must choose a relational database management system (RDBMS) and create the database in accordance with your selected RDBMS.

Before you begin

Choose which Relational Database Management System (RDBMS) you want to use for the data store. Unless you are using the embedded Apache Derby provider, create the database before you create a messaging engine. Make a note of the database parameters that you need for configuring the data source. For more information, see “Configuring a JDBC data source for a messaging engine” on page 1960.

Procedure

Refer to the documentation for your chosen RDBMS for information about how to create a database. The default database for a data store is an embedded Apache Derby database. If you have chosen to configure the bus member to use a data store with default settings, it can only be a server. Unless the data store database exists already, the messaging engine creates the database automatically when the messaging engine makes its initial connection.

Sybase tips:

- Ensure that you create the database server with a page size of at least 4 KB.
- Ensure that you set the **lock scheme** property on your server to the value `datarows`. This avoids the possibility of a deadlock on the data store tables.
- Ensure that you set the **enable housekeeper GC** property on your server to the value `5`. This improves the ability of the server to reclaim redundant space within your database when it is under heavy load.
- Ensure that you select the **allow nulls by default** option for your database instance. This is required for the correct operation of the messaging engine.

Informix tip: The one-to-one relationship between a messaging engine and a data store means that every messaging engine must have its own database tables. If you are using the Informix RDBMS, configure a separate database instance for each messaging engine. Problems have been observed in this environment when the data stores for multiple messaging engines were configured to use separate schemas in the same database.

Creating users and schemas in the database:

After you have created a database, you create the schema in which all tables in the data store are held. Depending on which database you are using, you create one or more database user IDs to enable the messaging engine to access the data store tables.

Before you begin

Before you begin this task, you must first create the database for your messaging engine.

About this task

All the tables in the data store must be stored in the same schema. You can create more than one data store in a database, provided that you use a different schema name for each data store. Although every messaging engine uses the same table names, its relationship with the schema gives each messaging engine exclusive use of its own tables.

To connect to WebSphere Application Server, you must create at least one messaging engine user ID. The number of user IDs you need depends on the database you use:

- If you are using Derby, DB2, or Oracle, the messaging engine can be configured to create any additional schemas that might be required for other data stores. That is, if you only create one user, it can have one to many relationships with the schemas in the database. See “Configuring a messaging engine data store to use a data source” on page 1899 for details.
- For all other types of databases, the schemas must be created before starting the messaging engines that depend on them.

If a database user ID can be configured to use multiple schemas, then only that user ID is needed for all messaging engines. Otherwise the user ID is restricted to using tables in its own schema. In this case there can be only one user ID per schema.

Procedure

1. Create the users and schema in accordance with the documentation for your chosen relational database management system (RDBMS). With DB2 databases, you create users and schema in separate steps. With the other databases, there is a one-to-one relationship between a schema and a user.
2. Ensure that the messaging engine user ID has the privileges required to enable the messaging engine to access the data store tables and, if required, create the data store tables automatically. See “Database privileges” on page 1967.

Creating data store tables:

When you create the data store tables, you have two options. You can either choose that WebSphere Application Server does it automatically, or that the database administrator does it manually.

Before you begin

Before you begin this task, decide whether you want WebSphere Application Server to create the tables automatically, or whether you want your database administrator to create the tables.

DB2 for z/OS restriction: The option for WebSphere Application Server to create the tables is not available with DB2 for z/OS. Refer to “Generating the DDL statements needed to create data store tables manually” on page 1960 if you use DB2 for z/OS.

Procedure

- If you want WebSphere Application Server to create the tables, complete the following steps:

1. Ensure that WebSphere Application Server has sufficient authority to create tables and indexes. For more information about the privileges required for your chosen database, see “Database privileges” on page 1967.
2. When you configure your messaging engine data store, ensure that the **Create tables** option is selected so that the messaging engine creates the tables in its chosen schema. For more information, see “Configuring a messaging engine data store to use a data source” on page 1899.

   DB2 for z/OS restriction: Do not select **Create tables**, otherwise an exception will be thrown when WebSphere Application Server attempts to create the tables.

- If you want your database administrator to create the database tables manually, use the sibDDLGenerator command to generate the DDL statements that the database administrator needs to create the tables for the messaging engine data store. For further information, see “Generating the DDL statements needed to create data store tables manually.”

Generating the DDL statements needed to create data store tables manually:

To enable your database administrator to create the data store tables manually, you must generate data definition language (DDL) statements.

Before you begin

Before you start this task, review the information in Configuration planning for a messaging engine to use a data store, and ensure that your database administrator has taken any appropriate action.

About this task

Use the sibDDLGenerator command to generate the DDL statements that the database administrator needs to create the tables for the messaging engine data store.

Procedure

1. At a command prompt, issue the sibDDLGenerator command and redirect the output to a file. Refer to “sibDDLGenerator command” on page 2425 for a description of the sibDDLGenerator command.

   Important: If you want to process the DDL statements with a command line processor that requires the statements to conform to a specific format, use the optional parameters that control the format of the DDL statements. For example, if each statement must end with a semicolon, use **-statementend** *;*

   To access the IBM i command line, or run an IBM i command line program, use the STRQSH command to start a Qshell session. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.
2. Send the output file to your database administrator to process the DDL statements that are generated. The DDL statements can be ported across operating systems, for example, you can generate the DDL statements on a machine running the Windows operating system and then run them on a machine running the z/OS operating system.

   Attention:

   - Your database administrator can modify the DDL statements, but *must not* modify the table names or the column names in any way because doing so might prevent the messaging engine from starting.
   - If the DDL statements are to be run on the z/OS operating system, your database administrator must change the VCAT name in the first line of the DDL statements (the create storage group statement) to a valid high-level qualifier for their system.

Configuring a JDBC data source for a messaging engine:

If you are using a data store for a messaging engine, the messaging engine uses an instance of JDBC data source to interact with the database containing the data store.

Before you begin

Restriction: When you are configuring a service integration bus member to use a data store, be aware that using a type 2 JDBC driver for the data store is not supported for configurations where WebSphere MQ server definitions are also used. If your configuration includes WebSphere MQ server definitions and you are using a data store, you must use type 4 JDBC drivers.

Apache Derby Tip: When you create a new Network Attached Apache Derby data store, by default you get a blank authentication alias.

Oracle Tip: Use the Oracle 10g thin driver for the service integration data store. This driver is compatible with earlier versions of Oracle.

About this task

Each messaging engine has its own file store or data store. If the data store is chosen the messaging engine uses an instance of a JDBC data source to interact with the database that contains the data store for that messaging engine.

Use the administrative console to set the data source configuration parameters. Note that your choice of relational database management system (RDBMS) determines the parameters you set.

Procedure

1. Create a JDBC provider. See “Configuring a JDBC provider using the administrative console” on page 273. Under **General Properties**, in the **Select the implementation type** field, ensure that you select *Connection pool data source*.

   For information about the settings for your chosen RDBMS, see “Data source minimum required settings, by vendor” on page 253.
2. Create a data source for the JDBC provider. Refer to “Configuring a data source using the administrative console” on page 280.
   a. Under **Additional Properties**, ensure that you select *Data sources*.
   b. Configure the connection pool for the JDBC data source. Set the **Maximum connections** to the number of connections you require, for example, at least 50. The default number of connections is 10. For more information, see ../ae/tjm0230_.dita.
3. Test the connection by using the test connection service provided for validating data source configurations See “Test connection service” on page 330.

Configuring a messaging engine data store to use a data source:

After configuring a JDBC data source, you can configure a messaging engine data store to use the data source.

Before you begin

To complete this task, you must have chosen or created a bus and a messaging engine, and the messaging engine must specify data store as its message store type.

You must also have configured a data source, as described in “Creating the database, schema and user ID for a messaging engine” on page 1899.

About this task

A messaging engine uses an instance of a JDBC data source to interact with the database that contains the data store for that messaging engine.

Use the WebSphere Application Server administrative console to set the data store configuration parameters.

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name***.
2. Check that the **Message store type** is *Data store*.
3. Click **[Additional Properties] Message store**. The data store configuration detail panel is displayed.
4. Specify the following data store configuration details:

   Data source JNDI name
   Type the JNDI name of the data source that provides access to database that holds the data store.

   Schema name
   Type the name of the database schema that contains the tables used by the data store.

   General tip: The schema name is usually the same as the user ID that is declared in the authentication alias. With some databases, for example DB2, you can provide an alternative schema name. For more information about the relationship between users and schema, refer to the documentation for your chosen RDBMS.

   Informix tip: When you configure your messaging engine to use an Informix database, you must specify the schema name in lowercase letters.

   When it is starting, a messaging engine that uses a data store checks to see if its data store exists. If the **Create tables** option is selected for the configuration, the messaging engine creates the tables in its chosen schema.

   The **Schema name** field is optional. If you require a schema name, consider the following:

   - The default schema name is `IBMWSSIB`.
   - If you delete the text so that field is blank, the messaging engine takes the user id defined in the authentication alias to be the schema name.
   - If you define a schema name explicitly, that schema name is used by the messaging engine.
   - If there are multiple messaging engines, you must configure each messaging engine to use a unique schema, otherwise FFDC error messages stating that `Connection cannot be provided as Datasource has been disabled!` might appear. This applies to DB2 in particular.

   Authentication alias
   Select the authentication alias that enables access to the data source.

   Apache Derby Tip: When you create a new Network Attached Apache Derby data store, by default you get a blank authentication alias.

   Create tables
   Select the check box if you want WebSphere Application Server to create the database tables automatically.

   Note: The user ID that the messaging engine uses to connect to the data source must have sufficient authority to create the database tables and indexes.

DB2 for z/OS restriction: Do not select **Create tables** if you are using DB2 for z/OS, otherwise an exception will be thrown when WebSphere Application Server attempts to create the tables.

Number of tables for permanent objects
Permanent tables contain persistent objects for the data store.

Note: You can only increase the number of permanent tables, not decrease them.

Number of tables for temporary objects
Temporary tables contain nonpersistent objects that have been saved to the data store to reduce the messaging engine memory requirement.

Note: You can only increase the number of temporary tables, not decrease them.

Backing up and restoring a messaging engine data store

Administering the data store for a messaging engine includes restoring and backing up the data store.

About this task

- "Backing up a data store"
- "Restoring a data store"

Backing up a data store:

Backing up a data store enables you to restore the data store from the backup if a failure occurs that cannot be dealt with by the system.

About this task

To back up the tables that comprise a data store, refer to the documentation for your chosen database.

Note: If your relational database management system (RDBMS) is DB2, and it is being used as the persistent data store, the backup process can use the suspended I/O feature of DB2. With other databases that do not possess this capability, the backup might present a longer interruption to service, or require that the messaging engine is stopped while the backup is made. If you attempt to back up the data store for a messaging engine that is still running, you might lose or corrupt important data.

Procedure

1. Unless your backup process uses the suspended I/O feature of DB2 stop the messaging engine.
2. Back up the data store in accordance with the documentation for your chosen database. Include the tables described in "Data store tables" on page 1967.

Restoring a data store:

If a failure occurs that cannot be dealt with by the system, you can restore the data store from a backup, if you are regularly backing up the data store tables.

About this task

To restore the tables that comprise a data store, refer to the documentation for your chosen database.

Note: You must stop the messaging engine before restoring the data store. If you attempt to restore the data store for a messaging engine that is still running, you might lose or corrupt important data.

Procedure

Restore the data store in accordance with the documentation for your chosen database. Include the tables described in “Data store tables” on page 1967.

Emptying the data store for a messaging engine

By emptying the data store of a messaging engine, you can discard persistent operating information without deleting the messaging engine.

About this task

Persistent operating information for a messaging engine is stored as persistent messages and associated information about message delivery and transmission. To discard this information without deleting the messaging engine and its destinations from the WebSphere Application Server configuration, you empty the messaging engine data store.

CAUTION:

- **When you empty the data store, all of the persistent messages that were held are lost and any destinations that you created continue to exist.**
- **Be very careful to completely empty the data store. You get unpredictable behavior if the data store is only partially emptied.**

Procedure

1. Ensure that the messaging engine and application server are stopped.
2. Empty the data store. There are several ways to achieve this:
   a. If you are using the **embedded Derby** database (as the default data store does) and the database contains just the tables for the messaging engine data store, delete the files that the database uses. This deletes the database. When the messaging engine is next started, it creates an empty database to replace it.

      Important: If you are using the same database for application data, you must instead empty the data store tables as described for any other RDBMS in a subsequent step.

      1) Find the database data directory in your file system. The name of the directory that contains the files used by the database is the same as the name of the database in the configuration of the JDBC data source used by the messaging engine data store. By default, this is

         `${USER_INSTALL_ROOT}/profiles/dmgr/databases/com.ibm.ws.sib/messagingEngineName`

      2) Delete the directory. If you have configured a separate log directory for your Derby database, delete this too. If you find that you cannot delete the files, confirm that the application server is also stopped (if you stop the messaging engine but not the application server, you cannot delete the files).

   b. If you are using the **Derby Network Server** database, use a similar procedure but also stop Derby Network Server before you delete the files. You must restart Derby Network Server before starting the messaging engine.

   c. If you are using any other RDBMS, empty the data store tables by using the administration tools of your RDBMS. You can either remove all data from the tables, or drop and recreate the tables.

      Most RDBMS support the TRUNCATE TABLE statement that removes all data from the tables. This is the preferred way of emptying the data store tables because it leaves the tables and their authorizations intact.

      If you RDBMS does not support the TRUNCATE TABLE statement (for example, DB2 does not), you can use the DELETE statement to delete all of the rows from all of the tables. However, if the tables contain a lot of data, this might not be practical because of resource limitations in the RDBMS. In this case, drop the tables and recreate them with the required indices and authorities.

Tip: If you have enabled WebSphere Application Server to create the data store tables, you can drop the tables by using the DROP TABLE statement. When the messaging engine is next started, it creates empty tables to replace them. If you have not enabled WebSphere Application Server (base) to create the data store tables, you must recreate the tables that you drop before you start the messaging engine.

Tip: You can use the **-drop** option of the sibDDLGenerator command to generate DDL to drop the tables.

Optional: If you have deleted the messaging engine by removing it from the bus, you can now recreate it.

3. Start the messaging engine and the application server.

Sharing connections to benefit from one-phase commit optimization

In some circumstances, you can configure your JMS application to share the JDBC connection that a messaging engine uses. Sharing connections enables you to use one-phase commit optimization. This can improve the performance of your application.

About this task

Messaging engines store persistent data in a database, and use a JDBC data source to interact with that database. Some JMS applications also store persistent data in a database, for example if the application uses entity enterprise beans. Typically, such applications use two-phase commit transactions to coordinate updates to the JMS and JDBC resources involved.

You can configure your application to share the JDBC connection used by a messaging engine, which enables you to use one-phase commit transactions and improve the performance of your application. You can benefit from the one-phase commit optimization in the following circumstances:

- Your application must use the assured persistent reliability attribute for its JMS messages.
- Your application must use container-managed persistence (CMP) entity beans that are bound to the same JDBC data source that the messaging engine uses for its data store.

Restriction: You cannot benefit from the one-phase commit optimization in the following circumstances:

- If your application uses a reliability attribute other than assured persistent for its JMS messages.
- If your application uses BMP entity beans, or JDBC clients.
- If your application uses DB2 High Availability Disaster Recovery (HADR).

Before you configure your system, ensure that you consider all of the components of your Java EE application that might be affected by one-phase commits.

Procedure

1. Select the assured persistent reliability attribute for your JMS messages.
2. Deploy all CMP enterprise beans involved in one-phase commit transactions with **res-auth** set to `Container`.
3. Deploy all CMP enterprise beans involved in one-phase commit transactions where **AccessIntent** maps to a transaction isolation level of `JDBC Read Committed`. You can choose any of the following values for **AccessIntent**:
   - `WSOptimisticUpdate`
   - `WSOptimisticRead`
   - `WSPessimisticUpdate-NoCollisions`

Oracle tip: All values for **AccessIntent**, except `WSPessimisticUpdateExclusive`, map to the `JDBC Read Committed` transaction isolation level.

DB2 tip: You can use any value for **AccessIntent**, because WebSphere Application Server exploits the DB2 dynamic transaction isolation level support.

4. Ensure that you use the same authentication alias for both your CMP enterprise beans and the messaging engine data store.
5. When you configure your JDBC data source, ensure that you select the **Use data source for CMP beans** option.
6. Set the value of the JDBC data source custom property **jmsOnePhaseOptimization** to `true`.
7. Use the JMS connection factory or activation specification panels to select the **Share data source with CMP** option.
8. When you create your JDBC provider and set the **Select the implementation type** field, ensure that you select `Connection pool data source` (and not `XA data source`).

Configuring messaging engine and server behavior when a data store connection is lost

If the connection between a running messaging engine and its data store is lost, either due to a failure or because you stop the database for maintenance, you can ensure that the messaging engine functions correctly after the connection is restored, by configuring the server to restart automatically.

About this task

The behavior described in this topic occurs only if the messaging engine is running and has established exclusive locks on its data store.

By setting the sib.msgstore.jdbcFailoverOnDBConnectionLoss custom property on a messaging engine, you can determine the behavior of the messaging engine and its hosting server in the event that the connection to the data store is lost.

Table 211. The behavior that is determined by the sib.msgstore.jdbcFailoverOnDBConnectionLoss custom property.. The first column of the table lists the sib.msgstore.jdbcFailoverOnDBConnectionLoss custom property values. The second column explains the behavior of the messaging engine when the data store connection is lost.

Property value	Behavior when the data store connection is lost
true (default)	The server shuts down and must be manually restarted.
false	The messaging engine continues to run and accept work, and periodically attempts to regain the connection to the data store. If work continues to be submitted to the messaging engine while the data store is unavailable, the results can be unpredictable, and the messaging engine can be in an inconsistent state when the data store connection is restored. **Note:** If work continues to be submitted to the messaging engine, even nonpersistent messaging can fail because the messaging engine might need to use the data store, for example to allocate a unique ID to a message, or to move nonpersistent messages out of memory.

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Custom properties** to navigate to the custom properties panel for the messaging engine.
2. Click **New**.
3. Type `sib.msgstore.jdbcFailoverOnDBConnectionLoss` in the Name field and `true` in the Value field.
4. Click **OK**.
5. Save your changes to the master configuration.
6. Restart the application server.

Results

If the connection between the messaging engine and its data store is lost, the application server that is hosting the messaging engine shuts down.

What to do next

After a server restart, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines** to view the status of the messaging engine. Check that the messaging engine has been restarted and is running.

You might want to tune your system so that the loss of the database connection is detected quickly, and the messaging engine waits for a reasonable amount of time for the data store to become available again before attempting to start on another server.

Data store tables

A data store uses a relational database management system (RDBMS), working through JDBC, to store data as rows in a set of tables. This data is important when you are backing up or restoring a data store.

The following table summarizes the purpose of each data store table.

Table name	Purpose
SIBOWNER	Ensures exclusive access to the data store by an active messaging engine.
SIBOWNERO	Used for locking the data store. This table stores no data in its one EMPTY_COLUMN column.
SIBCLASSMAP	Catalogs the different object types in the data store.
SIBLISTING	Catalogs the SIB*nnn* tables.
SIBXACTS	Maintains the status of active two-phase commit transactions.
SIBKEYS	Assigns unique identifiers to objects in the messaging engine.
SIB*nnn*, where *nnn* is a number	Contains persistent objects such as messages and subscription information. These tables hold both persistent and nonpersistent objects, using separate tables for the different types of data.

Note: The SIBOWNERO table was introduced for WebSphere Application Server Version 7.0 and must be created when you are migrating to WebSphere Application Server Version 7.0 or later from an earlier version of WebSphere Application Server. See ../ae/tjm0004_.dita for things to consider when you are migrating a messaging engine based on a data store.

Database privileges

In order for a messaging engine to use its data store, the database user ID that the messaging engine uses must have sufficient privilege to enable the messaging engine to access the data store tables. If you want the messaging engine to create the data store tables automatically, the messaging engine user ID requires additional privileges.

The following table describes the database privileges that the messaging engine user ID requires to access the data store and create the data store tables.

Database management system	Minimum privilege required for the messaging engine to use the data store tables	Additional privilege required for the messaging engine to create the data store tables
DB2	The messaging engine user ID needs **SELECT**, **INSERT**, **UPDATE**, and **DELETE** privileges on the tables.	The messaging engine user ID needs **CREATETAB** authority on the database and **USE** privilege on the table space as well as **CREATEIN** privilege on the schema.
Oracle	The messaging engine user ID needs at least **SESSION** privilege to connect to the database. If the same user ID owns both the data store schema and the messaging engine that is connecting to the database, the messaging engine has sufficient privilege to manipulate the tables. Otherwise, the messaging engine needs **SELECT**, **INSERT**, **UPDATE** and **DELETE** object privileges on the tables that comprise the data store, and **DROP ANY TABLE** system privilege to enable use of the TRUNCATE TABLE statement.	The messaging engine user ID requires sufficient privilege to create relational tables and indexes in the data store schema. The messaging engine also requires a space quota in the default tablespace of the owner of that schema.
SQL Server	Configure the SQL Server for **SQL Server and Windows** authentication. This allows authentication to be based on an SQL server login ID and password. The messaging engine user ID can be the owner of the tables, or be a member of a group that has sufficient authority to issue TRUNCATE TABLE statements.	The messaging engine user ID needs **CREATE TABLE** statement privilege.
Sybase	The messaging engine user ID can be the owner of the tables, or can be a member of a group that has sufficient authority to issue TRUNCATE TABLE statements.	The messaging engine user ID needs **CREATE TABLE** permission.
Informix	The messaging engine user ID must have **CONNECT** privilege on the database. It must also have **SELECT**, **INSERT**, **UPDATE** and **DELETE** authority on the tables.	The messaging engine user ID must have **RESOURCE** privilege on the database.
Derby	If user authentication is enabled, you must authorize the messaging engine user ID to access the database. **Remember:** The default database that is generated by the messaging engine has no security mechanisms enabled.	You need no additional privileges.

If you do not grant TRUNCATE TABLE authority to the database user ID, you can force the messaging engine to delete rows individually instead of truncating the table. To force the messaging engine to delete rows individually, set **sib.msgstore.jdbcUseDeleteInsteadOfTruncateAtStartup** to true as a custom property of the messaging engine.

Note: If you use DELETE instead of TRUNCATE, startup is slower when there are many non-persistent messages in the datastore.

Avoiding message store errors when creating a messaging engine

Using different combinations of parameters can create a file store or a data store according your requirements. The outcome varies in server and cluster scopes.

Server scope

When you add a server as a new bus member take note of the following:

- If you do not specify the type of message store, then a file store is created by default. If you set **Create default data source** to *True* or supply a **Data source JNDI name**, then a data store is created.
- If you choose to use a **file store**, then only file store attributes are presented to be specified. For example the **log file directory**.
- If you choose to use a **data store**:
  - Only data store attributes are presented to be specified. For example the **Data source JNDI name**.
  - If you set **Create default data source** to *False* then you must specify a **Data source JNDI name**.

For more information about file store and data store refer to “Administering message stores” on page 1953.

Cluster scope

When you add a cluster as a new bus member take note of the following:

- If you choose to use a **file store**:
  - You must not use the default log file, permanent store file and temporary store file directories because they are not suitable for cluster engines.
  - You must specify the **Log directory**, **Permanent store directory** and **Temporary store directory** to be at locations that all members of the cluster can access on your file system.
- If you choose to use a **data store**:
  - You must not use the default data source because it is not suitable for cluster engines.
  - You must specify the **Data source JNDI name** of an existing data source.

For more information about file store and data store refer to “Administering message stores” on page 1953.

Avoiding errors when creating a messaging engine with a file store or a data store by using the wsadmin tool

Using different combinations of parameters can create a file store or a data store according your requirements. The outcome varies in server and cluster scopes.

Server scope

When you add a server as a new bus member by using administrative commands (by specifying `createSIBEngine -server`) take note of the following:

- If you do not specify the type of message store, then a file store is created by default. If you specify `-createDefaultDatasource` or `-dataSourceJndiName`, then a data store is created.
- If you choose to use a **file store** (by specifying `-filestore`), then only file store attributes can be specified.
- If you choose to use a **data store** (by specifying `-datastore`):
  - Only data store attributes can be specified.
  - If you set `-createDefaultDatasource` to `False` then you must specify `-dataSourceJndiName`.

Cluster scope

When you add a cluster as a new bus member by using administrative commands (by specifying `createSIBEngine -cluster`) take note of the following:

- If you choose to use a **file store** (by specifying `-filestore`):

  - You must not use the default log file, permanent store file and temporary store file directories because they are not suitable for cluster engines.
  - You must specify `-logDirectory`, `-permanentStoreDirectory` and `-temporaryStoreDirectory` to be at locations that all members of the cluster can access on your file system.
- If you choose to use a **data store** (by specifying `-datastore`):
  - You must not specify `-createDefaultDatasource` because the default data source is not suitable for cluster engines.
  - You must specify `-dataSourceJndiName`, giving the name of an existing data source.

Administering bus destinations

Configure service integration bus destinations, so that applications can attach to them to exchange messages as producers, consumers, or both.

Procedure

- ../ae/cjo_learning.dita
- “Configuring bus destinations” on page 1905
- “Configuring message points” on page 1999
- “Managing messages on message points” on page 595
- “Administering durable subscriptions” on page 2002

Configuring bus destinations

Use the following tasks to configure permanent bus destinations on service integration buses.

About this task

The steps involved in configuring a bus destination depend on the intended usage of the destination.

For example, the following figure shows a basic scenario based on an application using a JMS queue for point-to-point messaging. A producing application sends messages to a JMS queue from which a consuming application retrieves the messages. The JMS queue is assigned to a queue destination and its associated queue point, where messages are stored.

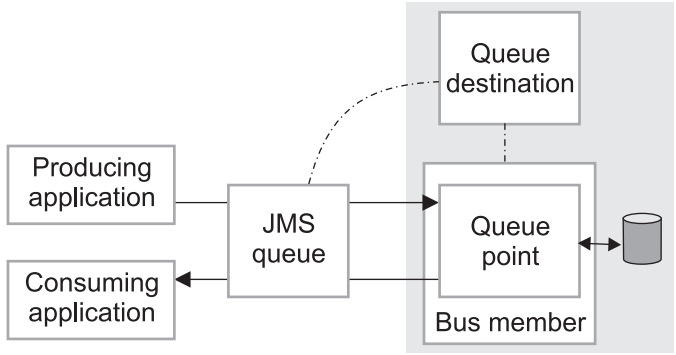


Figure 37. Application use of destinations - basic case

To enable the JMS applications to use a queue destination, you configure the following administrative destination objects:

1. A queue destination on a service integration bus. This configures the properties of the queue, such as the name, and associates the queue with one bus member (an application server). This also automatically configures, on the bus member, a queue point where messages for the queue are held and processed.
2. A JMS queue, which configures the name that applications can use to look up the queue in JNDI. The JMS queue encapsulates the name of the queue destination, as defined in the queue destination above, together with other properties to be used by applications.

After you have created a queue destination, you can optionally configure the queue point to override some properties configured on the queue destination. You can also undertake other configuration tasks on the destination and its queue point, and can act on the runtime view.

Each messaging engine has a default exception destination, named `_SYSTEM.Exception.Destinaton.`*`messaging_engine_name`*. This exception destination can be used to handle messages that cannot be delivered for all bus destinations that are localized to the messaging engine. Each bus destination can be configured with a non-default exception destination.

For more information about configuring bus destinations, see the following topics:
- Listing bus destinations
- Creating a bus destination
- Configuring qualities of service for a destination
- Deleting a non-topic space bus destination
- Deleting a topic space

Listing bus destinations

Use this task to display administrative lists of bus destinations for a service integration bus.

About this task

From the panel that lists bus destinations you can use the buttons provided to create, delete, or change the mediation of destinations, or can select a destination to browse details and options for that destination.

To list bus destinations on a bus, use the administrative console to complete the following steps.

Procedure

Click **Service integration -> Buses ->** ***bus_name*** **-> [Destination resources] Destinations**. This displays a list of the bus destinations on the bus.
Any temporary destination in the list is identified by the prefix `_Q` for temporary queues or `_T` for temporary topics.

What to do next

To browse or change the properties of a destination, click its name in the list displayed.

To act on one or more of the destination listed, select the check boxes next to the names of the destinations that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

Creating a bus destination

Use the following tasks to create a new bus destination on a service integration bus.

About this task

The steps involved in defining a new destination depend on the intended usage of the destination.

These are permanent destinations, with their properties defined by administrative resources, and are in addition to any temporary destinations created at runtime by the bus for applications.
- Defining a bus destination for JMS queues.
- Defining a bus destination for JMS topics.
- Defining an alias bus destination.
- Defining a new exception bus destination

Creating a queue for point-to-point messaging:

You can create a queue, which is a bus destination that represents a message queue and that is used for point-to-point messaging.

Before you begin

During this task you must specify the name of a bus member to which the bus destination is assigned. That bus member is to host the queue point for the new bus destination, and must already have been configured.

About this task

To define a new queue for point-to-point messaging, you specify only a minimum set of properties. You can change these properties and configure further properties after you complete this task.

To define a queue, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that is to provide the message point for the queue.
3. In the content pane, under Destination resources, click **Destinations**. A list of any existing bus destinations is displayed.
4. To create a destination, click **New**.
   a. On the Create new destination page, ensure that **Queue** is selected.
   b. Click **Next**.
   c. In the **Identifier** field, type the name that you want to give the queue destination for administrative purposes. Restrict the name to 48 characters or less, and restrict its character set to: numerics (0-9), period (.), forward slash (/), underscore (_), percent sign (%), uppercase A-Z, lowercase a-z (but there are restrictions on the use of lowercase letters for z/OS console support). On systems that use EBCDIC Katakana, you cannot use lowercase characters.
5. Optional: In the **Description** field, type a description of the destination, for administrative purposes.
6. Click **Next**.
7. On the Assign the queue to a bus member page, select the bus member that is to provide the queue point for the destination. The queue point is where the messages for the queue are held.
8. Click **Next**.
9. Optional: If the bus member is a WebSphere MQ server, set the WebSphere MQ queue point attributes:
   a. Specify a value in the **WebSphere MQ queue name filter** field, then click **Go**.

      The wizard automatically discovers available WebSphere MQ queues. However, some WebSphere MQ topologies have many thousands of queues defined to a queue manager. Use this filter to limit the number of queues that are listed.

The default filter value is an asterisk (*). If this value (or no value) is set then all queues, or all queues of a specific type (based on any queue type custom property that is set), are listed. Any other value that you specify must meet the following criteria:

- It must contain between 1 and 48 characters.
- It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

You can also use the wildcard character (*) with other text. For example, if you enter a value of PAYROLL*, then all available queues with names that start with PAYROLL are displayed.

b. Specify a WebSphere MQ queue name.

Select a queue name from the filtered list. If the list does not include the queue that you want, select the last entry in the list labeled **other, please specify**. A text entry box is displayed next to the drop-down list. Type the queue name into the text entry box.

If the queue is found on the remote WebSphere MQ system, the properties of the queue as defined within WebSphere MQ are displayed as read-only fields. This should help you to confirm that you have found the queue that you want, and that it is configured as you intend. If the queue is not found, these read-only fields are removed from view.

c. Specify the reliability levels that you require when inbound nonpersistent and inbound persistent WebSphere MQ messages are converted to service integration format messages. Applications receive messages direct from the specified WebSphere MQ queue, so in general the reliability level for a message is of no interest to the receiver because the message has already been delivered successfully. However, the message is converted to a service integration format message (and typically to a JMS format service integration message) as it is received, and this option specifies the reliability level for the service integration format message. For information about the available reliability levels, see “WebSphere MQ queue points [Settings]” on page 2241.

d. Specify whether you want WebSphere MQ to include an MQRFH2 message header when sending messages to the queue.

The MQRFH2 header stores service integration messaging information that does not have a corresponding WebSphere MQ message header field. When a message is sent to the destination, service integration instructs WebSphere MQ to write the message to the queue. This option specifies whether service integration instructs WebSphere MQ to write the message with an MQRFH2 header.

If the consumer of the message is a JMS application running in WebSphere MQ or service integration, or a WebSphere MQ XMS application, or a WebSphere MQ MQI application that expects an MQRFH2 header, select this option. If the consumer is a WebSphere MQ MQI application that does not expect an MQRFH2 header, do not select this option.

e. Click **Next**.

10. On the Confirm queue creation page, review the summary of actions.
    - To create the queue, click **Finish**.
    - If you want to change any of the properties that you have specified, click **Previous**, then change the properties on the preceding pages.
11. Save your changes to the master configuration.

What to do next

You can configure further properties of the queue, for example message reliability settings. See “Configuring bus destination properties” on page 1979. If you configure message reliability settings, remember that higher levels of reliability have a greater impact on performance.

By default, messages that cannot be delivered to the queue are sent to the default exception destination for the messaging engine that hosts the queue point. If you want to use a specific exception destination for

messages that cannot be delivered to this queue destination, that exception destination must be defined already. For more information about configuring exception destinations, see Configuring an exception bus destination.

If the queue is to be used for JMS point-to-point messaging, specify the queue identifier on a JMS queue, as described in “Configuring a queue for the default messaging provider” on page 522.

Creating a topic space for publish/subscribe messaging:

You can create a topic space, which is a bus destination that represents a set of “publish and subscribe” topics and that is used for publish/subscribe messaging.

About this task

You create a topic space when you deploy JMS applications that use publish/subscribe messaging.

To create a topic space, you specify only a minimum set of properties. You can change these properties and configure further properties after you complete this task.

To create a topic space, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that is to provide the publication points for the topic space.
3. In the content pane, under Destination resources, click **Destinations**. A list of any existing bus destinations is displayed.
4. To create a topic space, click **New**.
   a. On the Create new destination page, select **Topic space**.
   b. Click **Next**.
   c. In the **Identifier** field, type the name that you want to give the topic space for administrative purposes.
5. Optional: In the **Description** field, type a description of the destination, for administrative purposes.
6. Click **Next**.
7. On the Confirm topic space creation page, review the summary of actions.
   - To create the topic space, click **Finish**.
   - If you want to change any of the properties that you have specified, click **Previous**, then change the properties on the preceding pages.
8. Save your changes to the master configuration.

What to do next

You can configure further properties of the topic space, for example message reliability settings. See “Configuring bus destination properties” on page 1979. If you configure message reliability settings, remember that higher levels of reliability have a greater impact on performance.

By default, messages that cannot be delivered to the topic space are sent the default exception destination for the messaging engine that is publishing the message. If you want to use a specific exception destination for messages that cannot be delivered to this topic space, you must have already defined that exception destination. For more information about configuring exception destinations, see “Configuring exception destination processing for a bus destination” on page 1980.

If the topic space is to be used for JMS publish/subscribe messaging, specify the topic space identifier on a JMS topic as described in “Configuring a topic for the default messaging provider” on page 523.

Creating an alias destination on a bus:

You can create an alias destination on a service integration bus. An alias destination maps an alternative name for a queue destination or a topic space destination. Any alias destination properties that you set override the destination defaults.

About this task

An alias destination maps an alias bus name and destination name (identifier) to a target bus name and destination name where the bus name, or the destination name, or both, are different.

To define a new alias destination, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus on which the alias destination is to be created.
3. In the content pane, under Destination resources, click **Destinations**. A list of any existing bus destinations is displayed.
4. To create a destination, click **New**.
   a. On the Create new destination page, select **Alias**.
   b. Click **Next**.
   c. In the **Identifier** field, type the name of the destination for applications to use to refer to the alias destination.
5. Specify the following properties for the destination:

   Bus The name of the bus for applications to use to refer to the alias destination. If you leave this field empty, the name of the local bus is used.

   Target identifier
   The identifier of the target destination that this alias destination represents.

   If the alias destination targets a queue that is provided by WebSphere MQ, type the value as a concatenation of the queue name and the queue manager name, *queue_name@qmanager_name*; for example: `Queue1@Qmgr2`.

   Target bus
   The name of the bus that hosts the target destination. This can be the local bus, a foreign service integration bus, or a foreign bus that represents a WebSphere MQ network. The default is the name specified for the **Bus** property.
6. Optional: Specify the following properties for the destination. These will override the destination defaults.

   Description
   An optional description of the destination, for administrative purposes.

   Enable producers to override default reliability
   Controls the quality of service for message flows between producers and the destination. Select this option to use the quality of service specified by producers instead of the quality defined for the destination.

   Default reliability
   The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

INHERIT
Use the reliability configured on the target destination.

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Note: Higher levels of reliability have higher impacts on performance.

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

Maximum reliability
The maximum reliability of messages accepted by this destination.

Producers cannot send messages to this destination with a reliability higher than the value specified for this property.

INHERIT
Use the reliability configured on the target destination.

Best effort nonpersistent

Express nonpersistent

Reliable nonpersistent

Reliable persistent

Assured persistent

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

Send allowed
Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

INHERIT
Use the value configured on the target destination.

TRUE Producers can send messages to this destination.

FALSE
Producers cannot send messages to this destination.

Receive allowed
Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination.

INHERIT
Use the value configured on the target destination.

TRUE Consumers can receive messages from this destination.

FALSE
Consumers cannot receive messages from this destination.

Default forward routing path
The value to which a message's forward routing path will be set if the message contains no forward routing path. This identifies a sequential list of intermediary bus destinations that messages must pass through to reach a target bus destination. The format of the field is a list of line-delimited bus destinations specified as *bus*:*name*.

Delegate authorization check to target destination
Indicates which destination access role is checked when a user accesses the alias destination. When this option is selected, the destination access role of the target destination is checked. When this option is not selected, the destination access role of the alias destination is checked.

If you do not want to override the security of the target destination, select this option.

7. Click **Next**.
8. On the Confirm alias destination creation page, review the summary of actions.
   - To create the alias destination, click **Finish**.
   - To change any of the destination properties, click **Previous**, then change the properties on the preceding pages.
9. Save your changes to the master configuration.

What to do next

You can change properties or configure further properties of the alias destination. See “Configuring alias destination properties” on page 1985.

If you want to override security settings for the destination, see “Administering destination roles” on page 1991.

Creating a foreign destination on a bus:

You can create a foreign destination on a service integration bus. A foreign destination represents a destination that is defined in another bus (a foreign bus). You use a foreign destination when you need to override messaging defaults, security settings, or both for an individual destination on a foreign bus.

About this task

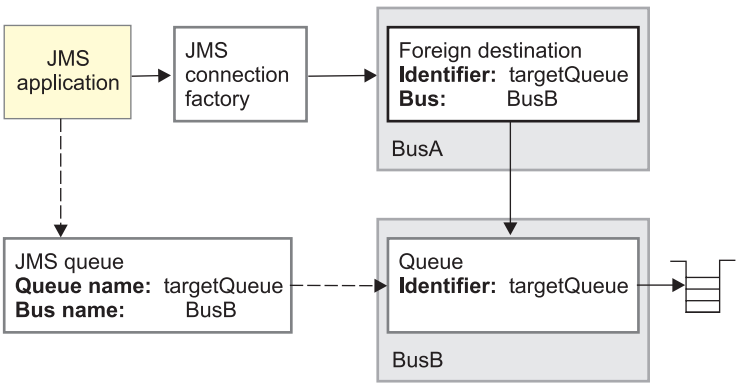


Figure 38. Creating a foreign destination on a service integration bus

This figure shows a foreign destination that points to a target destination on another bus. There is also a JMS connection factory and JMS queue that an application uses without being aware of the associated foreign destination.

The foreign destination encapsulates the name of the target destination that exists in the foreign bus (Identifier property) and the name of that foreign bus (Bus property). An application that is to use the foreign destination to exchange messages with the target destination must specify the Identifier and Bus properties.

For example, an administrator wants JMS applications to connect to one bus, BusA, and send messages to a JMS queue backed by a queue, targetQueue, on another bus, BusB. The administrator connects the buses, creates a foreign destination on BusA and sets the following properties on the foreign destination and JMS queue:

Table 212. Example property settings. The table provides an example of how the properties can be set when creating a foreign destination on a service integration bus. The first column of the table lists the JMS queue names. The second column contains the identifier and bus names of the foreign destination on a service integration bus, for example, BusA. The third column contains the identifier of the queue on a service integration bus, for example, BusB.

JMS queue	Foreign destination (on BusA)	Queue (on BusB)
Queue name targetQueue **Bus name** BusB	**Identifier** targetQueue **Bus** BusB	**Identifier** targetQueue

To define a new foreign destination, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus that you want to create the foreign destination on.
3. In the content pane, under Destination resources, click **Destinations**. A list of any existing bus destinations is displayed.
4. To create a destination, click **New**.
   a. On the Create new destination page, select **Foreign**.
   b. Click **Next**.
   c. In the **Identifier** field, type the name of the target destination that exists in the foreign bus. The Identifier must match the name of the target destination that exists in the foreign bus.
   d. In the **Bus** field, type the name of the foreign bus that hosts the target destination. On the bus where you are creating the foreign destination, a foreign bus connection that represents this foreign bus must be already defined.
5. Optional: Specify the following properties for the destination. These will override the destination defaults.

   Description
   An optional description of the destination, for administrative purposes.

   Enable producers to override default reliability
   Select this option to enable producers to override the default reliability that is set on the destination.

   Default reliability
   The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Note: Higher levels of reliability have higher impacts on performance.

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

Maximum reliability
The maximum reliability of messages accepted by this destination.

Best effort nonpersistent

Express nonpersistent

Reliable nonpersistent

Reliable persistent

Assured persistent

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

6. Click **Next**.
7. On the Confirm foreign destination creation page, review the summary of actions.
   - To create the foreign destination, click **Finish**.
   - If you want to change any of the destination properties, click **Previous**, then change the properties on the preceding pages.
8. Save your changes to the master configuration.

What to do next

Ensure that you have defined a foreign bus connection (to identify the target bus) and the target destination on that bus.

You can change properties or configure further properties of the foreign destination. See “Configuring bus destination properties.”

If you want to override security settings for the destination, see “Administering destination roles” on page 1991.

Configuring bus destination properties

You can view or change the configuration properties of a bus destination, that is, a queue, topic space, alias destination, or foreign destination.

About this task

For an overview of bus destinations, see ../ae/cjo_learning.dita.

To view or change the properties of a destination, use the administrative console to complete the following steps.

Procedure

1. From the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***. The configuration page for the type of destination you selected is displayed, that is, one of the following pages:
   - Queue settings
   - Topic space settings
   - Alias destination settings
   - Foreign destination settings
2. Change the destination properties to suit your needs. For details of the properties, see the relevant settings topic for the type of destination you selected.
3. Click **OK**.
4. Save your changes to the master configuration.

Configuring exception destination processing for a bus destination:

You can configure the exception destination processing for a queue destination or topic space destination. You can configure whether any undeliverable messages are rerouted to an exception destination, and whether to use a system default exception destination or configure a specific exception destination.

Before you begin

To configure a specific exception destination for a queue destination or topic space destination, the exception destination must exist. An exception destination must be a queue destination. See “Creating a queue for point-to-point messaging” on page 1972.

About this task

An exception destination for a queue destination or topic space destination is the destination for a message when a message cannot be delivered because the number of delivery attempts to a transactional consumer is exceeded.

You can configure an exception destination for a bus destination as one of the following:

- None. The bus destination does not use an exception destination and undeliverable messages are not rerouted to an exception destination. Attempts to redeliver the message continue, up to the maximum failed deliveries limit set for the bus destination. Then attempts to redeliver the message continue with a time interval between retry attempts. This interval is either the Default blocked destination retry interval of the messaging engine that is associated with this destination, or the Blocked retry timeout in milliseconds that is set for this destination.
- System. The bus destination uses the default exception destination. Messages that cannot be delivered to the bus destination are rerouted to the system default exception destination for the messaging engine that detects the problem: _SYSTEM.Exception.Destination.*messaging_engine_name*. This option is the default option.
- Specify. The bus destination uses the specified exception destination. If the bus destination cannot use this exception destination, it uses the system exception destination.

Note:

- You cannot configure exception destination processing for a bus; you must configure exception destination processing for each destination on the bus.
- Do not modify or delete the default system exception destination.
- If you use an exception destination for a bus destination, it can affect message ordering in the bus. For more information, see Message ordering.
- Best-effort messages are always discarded if they cannot be delivered to their target destination, that is, they never use an exception destination.
- Any report options that are set in the properties of a message also affect exception destination processing. Depending on the report options, a message might be discarded if it is not delivered.

To configure the exception destination processing for a bus destination, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane of the administrative console, click **Service integration -> Buses** to display a list of buses.
2. Select the bus with the destination for which you want to configure an exception destination.
3. In the **Configuration** tab, under **Destination resources**, click **Destinations** to display a list of destinations for this bus.
4. Select the name of the bus destination you require from the list. The details of that destination are displayed.
5. In the **Configuration** tab, under **General properties**, in the Exception destination section, use the radio buttons to configure the exception destination processing that this bus destination uses:
   - Select **None** to specify that the bus destination does not use an exception destination.
   - Select **System** to use the default exception destination.
   - Select **Specify** and enter an exception destination to configure the exception destination you require.
6. Optional: If you selected **None**, you can set the time interval to apply between retry attempts, after the maximum failed deliveries limit is reached, for this destination. Select **Override messaging engine blocked retry timeout default**, then enter the value you require in **Blocked retry timeout in milliseconds**. Otherwise, the value set for the Default blocked destination retry interval of the associated messaging engine is used.
7. Optional: To change the number of delivery attempts for a message, enter a value in **Maximum failed deliveries per message**. When the exception destination is configured as **None**, this is the number of delivery attempts before a time interval between retry attempts is applied. When the exception destination is configured as **System** or **Specify**, this is the number of delivery attempts for a message that is backed out and tried again before the message is sent to the exception destination.
8. Save your changes to the master configuration.

Results

You have configured the exception destination processing for a bus destination.

What to do next

You can also configure exception destination processing for a service integration bus link or WebSphere MQ link.

Controlling whether applications can send or receive messages for a bus destination:

You can prevent applications from either sending messages to, or receiving messages from, a destination. To do this you use the **Receive allowed**, **Send allowed**, and **Receive exclusive** properties of destinations

to control access to destinations. When you save changes to those properties, this affects open producers and consumers attached to localization points for that destination.

About this task

Use this task to change the configuration properties of a bus destination to control whether applications can send messages to, or receive messages from, a destination. For example, some destinations only exist in order to be associated with mediations; applications should not be able to put to or get from such a destination.

The changes that you make affect the configuration of a bus destination and when saved, are automatically applied to all message points for that destination. You can make the same changes to an individual destination localization point to control access to only that one point.

When you save changes that affect the access to a bus destination, this affects producers or consumers attached to message points for that destination. For each producer or consumer, any existing operations are allowed to complete (except for one case, as described in the next paragraph). The producer or consumer then undergoes a state change to conform to the destination, and subsequent operations will fail with an exception. The exception message indicates the specific reason for the exception; that is, that the destination no longer accepts sending or receiving of messages.

The only case where this behavior does not occur is the receiveWithWait() method. Blocking receives are cancelled when the state change to the consumer is made, and an exception is thrown. So, a receiveWithWait() method that is outstanding at the time of the configuration change is not allowed to complete, although the exception still occurs asynchronously with the configuration change.

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
2. Optional: Change one or more of the following properties:

   Receive allowed

   Clear this check box (setting to the option to false) to prevent messages from being received from message points for this destination. The effect depends on the type of destination:

   - Queue point. Any open consumers change state and an exception is thrown when the consumer requests a message.
   - Publication point. Any messages that have been published to the messaging engine for a publication point are stopped from proceeding either to local consumers or onwards to other messaging engines. Local consumers get the same exception as for a queue point.
   - Mediation point of a mediated destination. The bus stops the mediation instance that is running locally to the mediation point; other instances of the mediation running on other messaging engines continue as normal.

   In all cases, messages can continue to be sent, and accumulate on the destination localization point.

   Send allowed

   Clear this check box (setting the option to false) to prevent messages from being accepted onto the message points for this destination.

   - For a queue point of a non-mediated destination, or a mediation point of a mediated destination, new messages (from attached producers or forwarded from another destination) are redirected to any available message point. If no message points are available, then messages that have already been accepted onto the bus, and new messages from attached producers, are preserved by the bus until a message point becomes available. The only exception to this is the case of a destination with only one message point (queue point or mediation point depending on whether the destination is mediated or non-mediated), where

the producer is attached to the same messaging engine. In this case, an exception is thrown on each send call. The exception message indicates that the reason for the exception is that the only extant localization has been disabled for send. The producer remains open as normal, and any more send calls succeed if the **Send allowed** property of the localization is reselected (reset to true).

- For a queue point of a mediated destination, clearing this **Send allowed** property alters the behavior of the mediation instances that are sending to the destination in the same way as setting it to false on a non-mediated destination affects producing applications: Messages are sent instead to any alternative message point. If no localizations are available, are preserved by the bus until a message point becomes available. For any mediation instance (that is, on any server that has a mediation point), if the same server hosts a queue point, and that queue point is the only queue point for the destination, then the mediation changes to the “stopped on error” state.
- For a publication point, clearing this **Send allowed** property stops applications attached locally to the topic space from publishing messages. The send calls receive an exception, and the producer remains open.

Receive exclusive

If you select this check box (setting the option to `true`), then only one consumer can be attached to any message point. This property is particularly intended for use with queues, but can be used with any type of destination.

- For a queue, the bus chooses a queue point for each request to create a consumer. If the selected queue point already has an attached consumer, then the call fails with an exception (containing an exception message and linked exception that indicate the precise nature of the failure). There is no guarantee that all available queue points are used before the exception is thrown.
- For a topic space, only one consumer can attach to any given messaging engine.

If you change the **Receive exclusive** property from `false` to `true`, some consumers are selected to be the exclusive receivers in accordance with the rules above. All other consumers are detached from the destination, in the same way as is described above for a transition of the **Receive allowed** property from true to false.

3. Click **OK**.
4. Save your changes to the master configuration.

Specifying whether strict message order is preserved for a bus destination:

Use this task to change the configuration properties of a bus destination to control whether strict message order is preserved.

Before you begin

The changes made in this task affect the configuration of a bus destination. Before you begin this task, make sure that you fully understand the restrictions that apply to ordered destinations. For further details see Strict message ordering for bus destinations.

About this task

Although messages produced by a single producer to a single destination are seen by a consumer to that destination in the same order as they are produced, there are certain topologies and events, such as system failures, that can change the order of messages (see Message ordering). If strict message order is essential, you can configure a destination to preserve the order of the messages in a much more rigorous manner. Configuring a destination in this way, when used with restricted topologies, maintains message order in all circumstances (for further information, see Strict message ordering for bus destinations).

To preserve the order of the messages that are delivered to a destination, use the administrative console to complete the following steps:

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
2. Select the **Maintain strict message order** check box (setting the option to `true`).
3. Click **OK**.
4. Save your changes to the master configuration.

Results

An ordered destination has priority over several other configuration properties, such as the number of consumers that are allowed to attach to the destination. The system overrides these configuration properties at run time, and generates a system log warning. For more information about the configuration properties that the system overrides at run time, and the impact of including an ordered destination in your system, see Strict message ordering for bus destinations.

Specifying whether messages are forwarded to WebSphere MQ as JMS messages:

Use this task to create a destination context property that determines whether an MQRFH2 header is added to WebSphere MQ JMS messages produced by foreign or alias destinations.

Before you begin

This task assumes that you have created an alias bus destination or a foreign bus destination. For more information, see “Creating an alias destination on a bus” on page 1975 or “Creating a foreign destination on a bus” on page 1977.

About this task

Carry out this task if you want JMS messages produced by foreign bus destinations or alias bus destinations forwarded to WebSphere MQ. In this task, you define a destination context property called _MQRFH2Allowed that adds an MQRFH2 header to JMS messages. If you do not configure _MQRFH2Allowed, the default value is NO, and an MQRFH2 header is not added to the message. WebSphere Application Server Version 5.1 users may be aware of a WebSphere MQ flag called TARGCLIENT that was used to add RFH2 headers to JMS messages. For more information about the use of the MQRFH2 header in interoperating with WebSphere MQ, see Mapping the message body to and from WebSphere MQ format and Point-to-point messaging with a WebSphere MQ network.

To create and configure a context property called _MQRFH2Allowed on a foreign or alias destination, use the administrative console to complete the following steps:

Procedure

1. Display the context properties for a selected foreign or alias destination:
   a. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations**.
   b. Select the foreign or alias destination for which you want to create the context property.
   c. Under **Additional Properties**, click **Context properties**.
2. Click **New** to create a new context property.
3. Specify the following context information:

   Name Type the name `_MQRFH2Allowed`.

   Context type
   Select the information type `Boolean` in the drop-down list.

Context value
Type the value `true` .

4. Click **OK**.
5. Save your changes to the master configuration.

Results

An MQRFH2 header is added to all messages produced by the foreign or alias destination. This means that JMS messages will be forwarded to WebSphere MQ as JMS messages.

Configuring alias destination properties:

Use this task to browse or change the configuration properties of an alias destination on a service integration bus. An alias destination maps a bus name and destination name (identifier) to a target where the bus name, or the destination name, or both, are different. Any alias destination properties that you set override the destination defaults.

About this task

To browse or change the properties of an alias destination, use the administrative console to complete the following steps.

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the bus on which the alias destination was created.
3. In the content pane, under Destination resources, click **Destinations**. A list of existing bus destinations is displayed.
4. Select the alias destination that you want to configure.
5. You can view or change the following alias destination properties.

   Description
   An optional description for the bus destination, for administrative purposes.

   Target identifier
   The name of the destination for which this is an alias.

   Target bus
   The name of the bus on which the destination for which this is an alias exists.

   Use all target queue points
   This property is displayed only if the destination that the Target identifier and Target bus properties identifies has multiple queue points configured (that is, the destination is on a cluster bus member with multiple messaging engines) and the Target bus is the local bus.Whether to use all target queue points indicates whether all the queue points of the target queue can be used, including any queue points created after the alias is configured. When selected, the **Target queue points** menu is disabled.

   Unselected queue points
   A list of the queue points that are not addressable by the alias definition. The list is generated from the complete list of queue points for this queue. This list is enabled only when **Use all target queue points** is unchecked.

   Selected queue points
   A list of the queue points that are addressable by the alias definition. This list is enabled only when **Use all target queue points** is unchecked.
6. You can change the following alias destination properties to suit your needs. These values will override the destination defaults.

Enable producers to override default reliability
Controls the quality of service for message flows between producers and the destination. Select this option to use the quality of service specified by producers instead of the quality defined for the destination.

Default reliability
The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

INHERIT
Use the reliability configured on the target destination.

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Note: Higher levels of reliability have higher impacts on performance.

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

Maximum reliability
The maximum reliability of messages accepted by this destination.

Producers cannot send messages to this destination with a reliability higher than the value specified for this property.

INHERIT
Use the reliability configured on the target destination.

Best effort nonpersistent

Express nonpersistent

Reliable nonpersistent

Reliable persistent

Assured persistent

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

Default priority
The default priority for messages sent to the target destination, in the range 0 (lowest) through 9 (highest), or -1. The value -1 indicates that messages should use the default priority defined on the target destination. The default priority is used only if a message does not specify its own priority.

Send allowed
Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

INHERIT
Use the value configured on the target destination.

TRUE Producers can send messages to this destination.

FALSE
Producers cannot send messages to this destination.

Receive allowed
Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination.

INHERIT
Use the value configured on the target destination.

TRUE Consumers can receive messages from this destination.

FALSE
Consumers cannot receive messages from this destination.

Reply destination
The name of a destination to be appended to any non-empty reverse routing path of messages sent to this destination.

Reply destination bus
The bus on which the reply destination exists.

Default forward routing path
The value to which a message's forward routing path will be set if the message contains no forward routing path. This identifies a sequential list of intermediary bus destinations that messages must pass through to reach a target bus destination. The format of the field is a list of line-delimited bus destinations specified as *bus*:*name*.

Delegate authorization check to target destination
Indicates which destination access role is checked when a user accesses the alias destination. When this option is selected, the destination access role of the target destination is checked. When this option is not selected, the destination access role of the alias destination is checked.

Include an RFH2 message header when sending messages to WebSphere MQ
Indicates whether messages sent to WebSphere MQ include an RFH2 header. The RFH2 header stores additional information to that which is stored in theWebSphere MQ message header. This property applies when the target bus is a WebSphere MQ queue manager.

7. Click **OK**.
8. Save your changes to the master configuration.

What to do next

If you want to enable correct processing of messages, under Additional Properties click **Context properties**. This information adds to the context information derived from processing the message header.

If you want an expandable tree view of all the applications and messaging resources that reference the current destination, both directly and indirectly, under Related Items click **Application resources topology**. As many of the references as possible are resolved to links to the associated configuration panel for the referenced object.

Configuring mediations

Use this task to configure one or more mediations for a selected bus destination in a service integration bus.

About this task

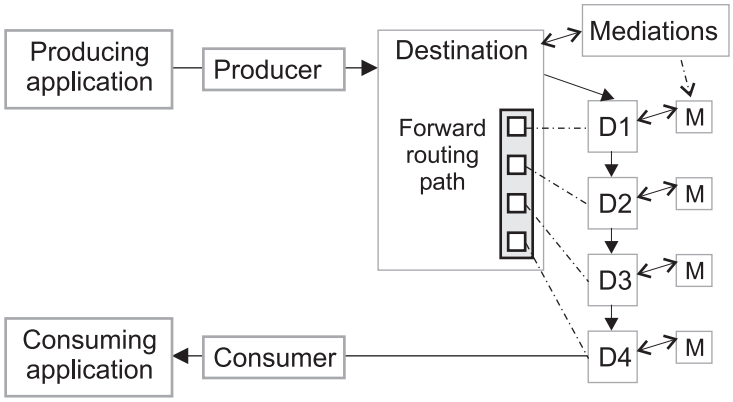
You can configure a destination with one or more mediations that refine how messages are handled by the destination, and can configure destination properties that affect how the mediations are used; for example, that they should run within a global transaction.

For more information about configuring mediations, see “Mediating a destination” on page 2017.

Configuring a destination forward routing path

Use this task to configure a forward routing path for a selected bus queue. The forward routing path identifies a sequence of bus destinations that a message should pass through after it has been delivered to the bus destination to which the producer is attached.

About this task



If producing and consuming applications both attach directly to the destination that is to host and process their messages, you do not have to use routing paths. You can still use mediations assigned to the destination to manipulate messages. However, if you want a more flexible architecture, you can assign mediations to several destinations then specify those destinations as the forward routing path for another destination. For example in the above diagram, on the destination called Destination, on the bus called myBus, you would set the Default forward routing path property to:

```
mybus:D1
mybus:D2
mybus:D3
mybus:D4
```

For each destination that should be visited by a reply message on its ways back to the producer application, you can specify the name of a *reply destination*, which is the next destination in the reverse routing path. If the same destinations are used for both the forward and return routing paths, you can configure the return routing path when you configure the forward routing path, as described in this topic. Otherwise, see “Configuring a destination reverse routing path” on page 1989.

To configure a forward routing path, you only have to change the properties of the destinations you want to use. This topic contains optional steps to create destinations, in case you have not already done so.

To configure a forward routing path for a destination, use the administrative console to complete the following steps:

Procedure

1. Optional: If you have not already done so, create the destination to which the producer applications attach. For this destination, you can create a queue or create a topic space.
2. Optional: If you have not already done so, create each destination in the path. Only the last destination in the path can be a topic space; all other destinations in the path must be queues.

   If the destination is also to be part of the return routing path for reply messages, specify the name of the next destination in the return routing path:

   a. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
   b. In the Reply destination bus field, type the name of the next destination in the return routing path.
   c. Save your changes to the master configuration.
3. Optional: If you have not already done so, assign mediations to the destinations in the path. You can include a destination without mediations in a routing path, to provide a future option to apply a mediation assigned to that destination.
4. Edit the properties of the destination to which the producer applications attach, to configure the forward routing path.

   a. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
   b. In the Default forward routing path field, type the names of the destinations in the path. Type a list of bus destinations, each on a separate line, of the form *bus_name*:*destination_name*

      Where

      bus_name
      Is the name of the service integration bus on which the destination is configured.

      destination_name
      is the name of the bus destination.

      For example:

      ```
      mybusA:queueA
      anobusB:queueB
      busC:queueC
      ```
   c. Save your changes to the master configuration.

Configuring a destination reverse routing path

Use this task to configure a reverse routing path between two bus destinations. The reverse routing path identifies the list of destinations to which a reply message should be sent when the consumer of a request message sends a reply message back to the producer of the reply message.

About this task

In the following figure, a producing application sends messages to a bus destination. Unknown to the application, the messages are then passed along a forward routing path of other bus destinations to a target destination where a consuming application retrieves the messages.

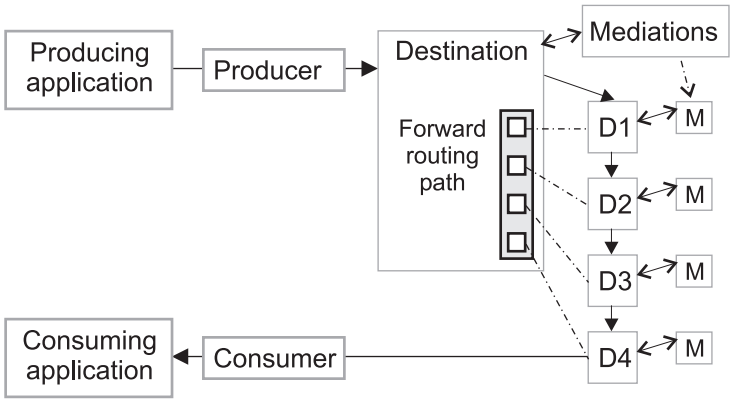


Figure 39. Example of a routing path

For each destination that is to be part of a reverse routing path back to the producer application, you can specify the name of a *reply destination*, which is the next destination in the reverse routing path. For example in the above diagram, if reply messages are to retrace the forward routing path back to the producing application, you would set reverse routing path as follows:

Destination	Reply destination bus
D4	D3
D3	D2
D2	D1
D1	Destination

You can specify a different reverse routing path to the forward routing path, to enable more mediations to be applied to reply messages, or to apply mediations in a different destination sequence.

If the same destinations are used for both the forward and reverse routing paths, you can configure the reverse routing path when you configure the forward routing path, as described in “Configuring a destination forward routing path” on page 1988. Otherwise, you can configure (or change) the reverse routing path as described in this topic.

To configure a reverse routing path, you only have to change the properties of the destinations you want to use. This topic contains optional steps to create destinations, in case you have not already done so.

To configure a reverse routing path for a destination, use the administrative console to complete the following steps:

Procedure

1. Optional: If you have not yet done so, create the destination to which the producer applications attaches. For this destination, you can create a queue or create a topic space.
2. Optional: If you have not already done so, create each destination in the path.
3. Optional: If you have not already done so, assign mediations to the destinations in the path. You can include a destination without mediations in a routing path, to provide a future option to apply a mediation assigned to that destination.
4. On each destination in the reverse routing path, specify the name of the next destination in the path:
   a. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
   b. In the Reply destination field, type the name of the next destination in the reverse routing path.

c. In the Reply destination bus field, type the name of the next destination in the reverse routing path.
d. Save your changes to the master configuration.

Configuring context properties for a bus destination

Use this task to configure context properties for a bus destination.

About this task

The context properties contribute to the mediation context that is used with the message header information to ensure that messages are processed correctly by a mediation assigned to this bus destination.

Note: Context properties on the destination take precedence over context properties on a mediation. For example, if a property with the same name is configured for a destination and a mediation, the property on the destination takes precedence.

To configure context information for a bus destination, use the administrative console to complete the following steps:

Procedure

1. Display the context properties for a selected bus destination:
   a. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
   b. Under **Additional Properties**, click **Context properties**.
2. Choose whether to create a new context property, configure an existing property, or delete one or more existing properties.
   - To create a new context property, click **New**.
   - To browse or change an existing context property, click the property name.
   - To delete one or more existing context properties, select the check box for each property you want to delete, then click **Delete**.
3. To create or change a context property, specify the following context information:

   Name Type a name for the context property.

   Context type
   Select the information type for the context property in the selection list.

   Context value
   Type a value for the context property.
4. Click **OK**.
5. Save your changes to the master configuration.

Administering destination roles

Service integration bus security uses role-based authorization. When messaging security is enabled, users and groups must have authority to undertake messaging operations, at a bus destination. By administering destination roles, you can control which users and groups can undertake operations at a bus destination, and the types of operations that they can perform.

About this task

You use the administrative console to administer users and groups in access roles for a destination. The access roles available for a destination depend on the type of destination. The table below lists the roles that you can assign for each destination type:

Table 213. Destination roles. The first column of the table contains the list of destination types. The second column contains the access roles that can be assigned for the destination types.

Destination type	Access roles
queue	sender, receiver, browser, creator
port	sender, receiver, browser, creator
webService	sender, receiver, browser, creator
topicSpace	sender, receiver
foreignDestination	sender
alias	sender, receiver, browser

In addition to controlling which users and groups have access to a specific local or foreign destination, you can also control the inheritance of access roles for a specific local destination. In this case, the default access roles that apply to all the destinations in the local bus namespace are added to any access roles that have been added for a specific destination.

Use the following tasks to administer destination roles.
- “listGroupsInDestinationRole command” on page 2396
- “addGroupToDestinationRole command” on page 2398
- “removeGroupFromDestinationRole command” on page 2400
- “listUsersInDestinationRole command” on page 2402
- “addUserToDestinationRole command” on page 2403
- “removeUserFromDestinationRole command” on page 2405
- “Defining destination defaults inheritance by using the wsadmin tool” on page 2417
- “Determining destination defaults inheritance by using the wsadmin tool” on page 2416

Adding users and groups to destination roles:

Service integration bus security uses role-based authorization. By adding users and groups to the destination roles for a secured bus, you can control which users and group members can undertake messaging operations at a bus destination.

Before you begin

Ensure that the following conditions are met:
- Security is enabled for the bus. For more information, see Securing buses.
- The users and groups that you want to add to destination roles must exist already in the user repository.

About this task

By adding users or groups to the destination role, you grant the users or groups authority to undertake the operation defined by the role at a selected destination. The destination roles are sender, receiver, browser, and creator, depending on the destination type.

In this task you use the administrative console Security wizard to retrieve selected users or groups from the user repository, and add them to destination roles for selected bus destinations.

Tip: To add a large number of users to destination roles, it is advisable to create a group in the user repository, and add the group to the destination roles.

Procedure
1. Start the administrative console.
2. Click **Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage destination access roles**. A list of the destinations defined for the selected bus is displayed in the Destinations panel.

3. Select one or more destination to work with:
   - Click a single destination name.
   - Select the check boxes next to multiple destination names, and then click **Manage Access Roles**.

   The Destination access roles panel is displayed. The information for each destination you have selected is displayed in a collapsed section.
4. Expand a destination header to list the users and groups that have been assigned to roles for this destination. You can verify that the user or group you want to add does not already have a role at this destination.
5. Click **Add** to start the Security wizard. The wizard takes you through the following steps to add selected users or groups to access roles for the expanded destination:
   a. Search for the users or groups that you want to add to access roles for the expanded destination:

      Users or Groups
      Select either **Users** or **Groups** to specify whether you want to grant access roles to users or groups.

      Search pattern
      This field is mandatory. Specify a search string that is matched against user IDs or group names in the user repository. Only user IDs or group names that match the search pattern are retrieved, subject to the maximum number of search results. Wildcard characters are allowed.

      Maximum number of search results to display
      This field is mandatory. Specify the maximum number of user IDs or group names you want the administrative console to display.

   b. Click **Next**. The wizard displays the users or groups in the user repository that match the information that you provided in the previous step.
   c. Select the check boxes next to the user IDs or group names that you want to add to access roles for the currently expanded destination, and click **Next**. A list of user IDs or group names that you can add to destination roles is displayed. Note that some users or groups might already be assigned to access roles for this destination.
   d. Select the appropriate access role icon for the user ID or group name that you want to add to the role at this destination. For example, select the **Receiver** icon for a user ID or group name that you want to add to the receiver role. The icon changes from  to  to show that you have added the user or group to the access role for the resource.
   e. Repeat the previous step to add more users or groups to access roles for the currently expanded destination, and then click **Next**. A summary of your access role assignments is displayed.
   f. Click **Previous** to review and change your assignments, if required.
   g. Click **Finish** to confirm your assignments.
6. Repeat steps 4 and 5 for each destination you want to work with.
7. Save your changes to the master configuration.

Results

The selected users and groups are added to the access roles for the currently expanded destination. The new access role assignments are displayed in the Destination access roles panel.

Example

A group called MyGroup receives messages from three queues, Queue 1, Queue 2, and Queue 3. If you want the group MyGroup to produce and consume messages at an additional destination, Queue 4, you add MyGroup to Queue 4, and then add MyGroup to the sender and receiver roles for Queue 4.

What to do next

Use the administrative console to complete other security administrative tasks.

Removing users and groups from destination roles:

Service integration bus security uses role-based authorization. By removing users and groups from the destination roles for a secured bus, you can prevent those users and group members from performing messaging operations on the bus.

About this task

When selected users and groups no longer require access to a destination, you can remove them from all the roles for that destination.

Procedure

1. Log into the administrative console.
2. Click **Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage destination access roles** A list of the destinations defined for the selected bus is displayed in the Destinations panel.
3. Select one or more destinations to work with:
   - Click a single destination name.
   - Select the check boxes next to multiple destination names, and then click **Manage Access Roles**.

   The Destination access roles panel is displayed. The information for each destination you have selected is displayed in a collapsed section.
4. Expand a destination header to list the users and groups that have been assigned to roles at this destination, and verify that the user or group that you want to remove has a role at this destination.
5. Select the users and groups that you want to remove from all role types at this destination, and click **Remove**.
6. Save your changes to the master configuration.

Results

The selected users and groups are removed from all role types at the selected destination. The Manage access roles for users and groups panel displays the updated role type assignments.

Example

The members of three groups, Group A, Group B, and Group C, belong to the sender role and the receiver role for two queue destination, Queue 1 and Queue 2. If Group B is no longer required to send and receive messages on Queue 2, you can use this task to remove Group B from all the role types on Queue 2.

What to do next

Use the administrative console to complete other security administrative tasks.

Listing users and groups in destination roles:

Service integration bus security uses role-based authorization. By listing the users and groups in the destination roles for a selected secured bus, you can find out which users and groups are authorized to access the bus, and its resources.

About this task

In this task you use the administrative console to list all the users and groups in destination roles for selected destinations. The list includes users and groups that have references in the service integration role-based configuration; it does not include all the users and groups that exist in the user repository. The permitted destination roles are sender, receiver, browser and creator, depending on the destination type. Icons are used in the administrative console to represent the roles to which users and groups have been assigned. For example, if the role type set icon () is displayed in the sender role for a group called Group 1, it means that Group 1 has been assigned to the sender role for a selected destination. For a complete description of all the icons used to represent role assignments in the administrative console, see "Access role assignments for bus security resources" on page 2429.

Procedure

1. Log into the administrative console.
2. Click **Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage destination access roles**. The Destinations panel lists all the destinations defined for the selected bus.
3. Select one or more destinations to work with:
   - Click the name of a single destination.
   - Select the check boxes next to multiple destinations, and click **Manage Access Roles**.

   The Destination access roles panel is displayed. The information for each selected destination is displayed in a collapsed section.
4. Expand a destination header.

Results

The Destination access roles panel lists the users and groups in access roles for the expanded destination.

What to do next

You can now administer the users and groups in destination roles at this destination.

Restoring default inheritance for a destination:

Service integration bus security uses role-based authorization. By default, all local destinations inherit access roles from the default resource. If default inheritance has been previously overridden, you can restore it for a selected destination.

Before you begin

Default inheritance has been overridden for a selected secured destination. For more information, see Overriding inheritance from the default resource for a destination.

About this task

If default inheritance has been overridden for a particular destination, you can restore it. In this task, you use the administrative console to restore the role type assignments from the default resource to a selected destination. A destination can only inherit access roles that are allowed for that particular type of destination. For example, a topic space can inherit the sender and receiver roles, but it cannot inherit the browser role. Inherited access roles are added to any existing access roles for the destination.

Procedure

1. Log into the administrative console.
2. Click **Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage destination access roles**. The Destinations panel lists all the destinations defined for the selected bus.
3. Select one or more destinations to work with:
   - Click the name of a single destination.
   - Select the check boxes next to multiple destinations, and click **Manage Access Roles**.

   The Destination access roles panel is displayed. The information for each selected destination is displayed in a collapsed section.
4. Expand a destination to list the users and groups that have been assigned to roles for this destination.
5. Select the **Inherit from default** check box.
6. Click **OK** to save your changes.
7. Save your changes to the master configuration.

Results

The role type assignments for the default resource are inherited by the selected destination. The Destination access roles panel displays the newly inherited default access roles for the destination, and any existing access roles.

Disabling inheritance from the default resource:

You can disable the inheritance of security access roles from the default resource for selected resources.

Before you begin

About this task

In this task, you use the administrative console to disable the inheritance of access roles from the default resource for selected resources. Disabling inheritance from the default access roles means that users or groups that have roles in the default access role no longer have access to this destination.

Use the administrative console to disable the inheritance of access roles from selected resources as follows:

Procedure

1. Use the administrative console to navigate to a list of the resources you want to work with for a specific bus. For example, to list all the known destinations for a selected bus, click **Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage destination access roles**. The Destination panel lists all the destinations defined on the selected bus.
2. Do one of the following to select one or more resources:
   - Click the name of a single resource.
   - Select the check boxes next to multiple resources, and click **Manage Access Roles**.

   The Access Roles panel displays access role information for each selected resource within a collapsed section.
3. Expand a resource header to list the users and groups that have been assigned to roles for this resource.
4. Clear the **Inherit from default** check box.
5. Click **OK** to save your changes. The role types for the default resource are removed from the selected resource, and the Access Roles panel is updated.

6. Save your changes to the master configuration.

Deleting a bus destination

Use these tasks to delete a bus destination from a service integration bus.

About this task

The steps required depend on the type of destination.

Deleting a non-topic space bus destination:

Use this task to delete, from a service integration bus, a bus destination that is not for a topic space.

Before you begin

You cannot delete bus destinations that are required by the system; for example, the default exception destination for a messaging engine: `_SYSTEM.Exception.Destination.`*`messaging_engine_name`*.

Before you can completely delete a bus destination, you must ensure that the destination message point is empty of all messages. A delete action is rejected if there are any messages on the message point that are committed, uncommitted, indoubt, or locked.

Any reliable or durable messages that are sent after the delete action, but before the close of producers, are sent to one of the following destinations:

- Another message point of the destination, if possible.
- The exception destination for the deleted destination.

Note: Any attached producers or consumers are closed asynchronously, and do not prevent the delete action. Any messages sent after the delete action, but before close of producers might not be discarded. Depending on the options set, the messages might be moved to the exception destination. For more information see “Configuring exception destination processing for a bus destination” on page 1980 and ../ae/cjo0004_.dita.

About this task

To delete a destination, use the administrative console to complete the following steps.

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations**.
2. Select the check box next to the destination name.
3. Check for and resolve any locked messages. Messages that are locked, as part of in-flight transactions or as indoubt messages, are not deleted.
4. To clear all messages from the destination, click **Clear**. This action deletes all of the messages that it can access. Messages that are locked, part of in-flight transactions or are in doubt are not deleted. If there are messages that cannot be deleted, they are skipped and the operation indicates that it was partially successful.
5. To delete the empty destination, click **Delete**.
6. To confirm that you want to delete the destination click **OK**.

Results

The bus destination is deleted.

What to do next

Monitor the exception queue for the messaging engine to which the destination was assigned, to see if there are any in-transit messages that were not handled by clearing the destination before it was deleted.

Deleting a topic space:

Use this task to delete, from a service integration bus, a topic space.

Before you begin

Before you can delete a topic space, you must ensure that the publication points are empty. A delete action is rejected if there are any indoubt or uncommitted publications for a publication point.

Note: Any attached producers are closed asynchronously, and do not prevent the delete action. However, any messages sent after the delete action, but before close of producers are discarded.

The service integration bus topic space is the primary messaging object upon which WS-Notification depends at run time. For information about the effect that deleting a topic space has upon new and existing WS-Notification applications, see Failures as a result of changes in topic space and topic namespace configurations.

About this task

To delete a topic space, use the administrative console to complete the following steps.

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations**.
2. Select the check box next to the destination name.
3. Check for and resolve any undelivered publications (indoubt or uncommitted) for the topic space. Messages that are locked, as part of in-flight transactions or as indoubt messages, are not deleted.
4. To delete the empty topic space, click **Delete**.
5. To confirm that you want to delete the topic space, click **OK**.
6. Save your changes to the master configuration.

Results

The topic space is deleted.

Resetting a destination

Use this task to reset a bus destination.

About this task

You might need to reset a destination if, for example, it has become corrupt. To reset a destination, use a wsadmin AdminControl command.

Note: If a destination becomes corrupt, a CWSIK0027E message is reported, which indicates a serious error. If you see this message, refer to Diagnosing problems (using diagnosis tools) for advice about gathering and reviewing information that might be useful in diagnosing this error. Also refer to the *MustGather: Collect troubleshooting data for a Service Integration Bus (SIB) problems* document on the Must gather website before opening a PMR.

Procedure

1. Set the name of the messaging engine by using a wsadmin AdminControl command of the following form:
   - Using Jython:
     ```
     me = AdminControl.queryNames("type=SIBMessagingEngine,name=messaging_engine_name,* " )
     ```
   - Using Jacl:
     ```
     set me [$AdminControl queryNames type=SIBMessagingEngine,name=messaging_engine_name,*]
     ```

   where *messaging_engine_name* is the name of the messaging engine.
2. Ensure that the messaging engine is running. Start the messaging engine if it is not already running. Read “Starting a messaging engine” on page 1951 for more information.
3. Reset the destination by using a wsadmin AdminControl command of the following form:
   - Using Jython:
     ```
     AdminControl.invoke(me, "resetDestination", ["destination"] )
     ```
   - Using Jacl:
     ```
     $AdminControl invoke $me resetDestination {"destination"}
     ```

   where *destination* is the name of the destination.
4. To complete the resetting of the destination, restart the server that is hosting the messaging engine on which the destination is localized.

Results

The bus destination is reset ready for use.

Managing bus destinations with administrative commands

You can use these commands to manage bus destinations.

About this task

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and JACL.

Procedure

1. Open a wsadmin command session in local mode For example:
   ```
   wsadmin -conntype none -lang jython
   ```

   Note: The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.
2. Type `AdminTask.command`

   Where *command* is the command format as indicated in the related reference topics; for example:
   ```
   wsadmin>AdminTask.listSIBDestinations(["-bus", "abus"])
   '(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBTopicSpace_1098181446388)
   (cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBQueue_1098181503600)
   (cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBQueue_1098184221748)'

   wsadmin>AdminTask.listSIBDestinations(["-bus", "abus", "-type", "TopicSpace"])
   '(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBTopicSpace_1098181446388)'
   ```

Configuring message points

Use the following tasks to configure message points on service integration buses.

About this task

A message point is the location (for example, queue point or publication point) at which messages for a bus destination are held in a messaging engine. The runtime state of the message point represents a runtime instance of the destination.

When you define a new bus destination, its message point is created. You can then configure some properties of the message point to override the general properties defined for the destination.

You can also administer the messages on the runtime view of the message point.
- “Configuring a message point”
- “Listing message points for a bus destination”
- “Listing message points for a messaging engine”

Listing message points for a messaging engine

Use this task to list the message points for bus destinations on a selected messaging engine.

About this task

To display a list of message points for a messaging engine, use the administrative console to complete the following steps.

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines**. This displays a list of messaging engines on the bus.
2. Click the name of the messaging engine.
3. Under Message Points, click the link for the type of point you want to list.

Results

A list of message points for the selected messaging engine is displayed in the content pane.

Listing message points for a bus destination

Use this task to list the message points for a selected bus destination.

About this task

To display a list of message points for a bus destination, use the administrative console to complete the following steps.

Procedure

1. Display the bus destination.
2. In the content pane, under Message Points, click the link for the type of point you want to list.

Results

A list of message points for the selected bus destination is displayed in the content pane.

Configuring a message point

Use this task to browse or change the configuration properties of a message point for a bus destination.

About this task

You can configure some properties of the message point to override the general properties defined for the destination.

To browse or change the properties of a message point, use the administrative console to complete the following steps:

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
2. Under Message Points, click the link for the type of message point:
   - **Queue points**, for a queue
   - **Publication points**, for a topic space
   - **Mediation points**, for a mediated queue or topic space
3. In the content pane, click the name of the message point.
4. Optional: Change the message point settings.
5. Click **OK**.
6. Save your changes to the master configuration.

Managing messages on message points

Use these tasks to list and act on runtime messages that exist on message points in a service integration bus.

About this task

You can list the message points for bus destinations and messaging engines, and list the messages on a selected message point. You can use the list of messages as part of a troubleshooting task to find messages that need to be deleted.

- "Listing messages on a message point" on page 595
- "Deleting messages on a message point" on page 596

Listing messages on a message point

Use this task to list the messages that exist on a message point for a selected bus destination or messaging engine.

About this task

To display a list of messages on a message point, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Buses**.
2. In the content pane, click the name of the service integration bus.
3. Optional: To list the message points for a bus destination, complete the following steps:
   a. In the content pane, under **Destination resources**, click **Destinations**.
   b. Click the destination name.
4. Optional: To list the message points for a messaging engine, complete the following steps:
   a. In the content pane, under **Topology**, click **Messaging engines**.
   b. Click the messaging engine name.
5. Under Additional Properties, click **Message points**. This displays a list of message points in the content pane.
6. Click the message point name. This displays the properties of the destination localization in the content pane.
7. Click the Runtime tab.

8. Under Additional Properties, click **Messages**.

Results

A list of messages on the selected message point is displayed in the content pane.

What to do next

You can select one or more messages to act on; for example, to display the message content, delete messages.

Deleting messages on a message point

Use this task to delete one or more messages that exist on a message point for a selected bus destination or messaging engine.

About this task

You should not usually have to delete messages on a message point. This task is intended as part of a troubleshooting procedure.

To delete one or messages on a message point, use the administrative console to complete the following steps:

Procedure

1. List the messages on the message point.
2. In the content pane, select the check box next to each message that you want to delete. Alternatively, you can select all messages in the list by clicking **Select all items**  .
3. Click **Delete**.

Results

The selected messages are removed from the list.

Administering durable subscriptions

Use the following tasks to display the durable subscriptions that exist, to enable a subscription to be changed, or to delete a subscription.

About this task

The default messaging provider supports the use of durable subscriptions to topics. This enables a subscriber to receive a copy of all messages published to a topic, even messages published during periods of time when the subscriber is not connected to the server.

If an application creates a durable subscription, it is added to the list that administrators can display and act upon by using the administrative console. Each durable subscription is created with a unique subscription identifier, *clientID##subName* where:

clientID
: The client identifier used to associate a connection and its objects with the messages maintained for applications (as clients of the JMS provider). You should use a naming convention that helps you identify the applications, in case you have to relate durable subscriptions to the associated applications for runtime administration. For more information about client identifiers, see section 4.3.2 of the JMS 1.1 specification.

subName
The JMS durable subscription name used to uniquely identify a durable subscription within a given client identifier. For more information about JMS durable subscription names, see section 6.11.1 of the JMS 1.1 specification.

For durable subscriptions created by message-driven beans, the subscription name value is set on the JMS activation specification. For other durable subscriptions, the value is set by the administrator on the JMS connection factory and by the JMS application on the createDurableSubscriber operation.

Note: The *server_name*-durableSubscriptions.ser file in the *WAS_HOME*/temp directory is used by the messaging service to keep track of durable subscriptions for message-driven beans. If you uninstall an application that contains a message-driven bean, this file is used to unsubscribe the durable subscription. If you have to delete the *WAS_HOME*/temp directory or other files in it, ensure that you preserve this file.

Procedure

- List durable subscriptions.
- Stop active subscribers for durable subscriptions.
- Enable providers to stream messages to cloned durable subscriptions.
- Delete durable subscriptions.

Listing subscriptions

Use this task to list the subscriptions that exist at runtime for a topic space.

About this task

The runtime state of a subscription is linked to the publication point for the messaging engine that is used to store messages delivered to the subscription. You can list subscriptions by first displaying the publication point, either from the messaging engine panel or the list of all publication points for a service integration bus.

To display a list of subscriptions for a publication point, use the administrative console to complete the following steps:

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations**. This displays any existing destinations in the content pane.
2. Click the name of the topic space.
3. Under Message Points, click **Publication points**. This displays the publication points for the topic space.
4. Click the name of the publication point.
5. Click the Runtime tab.
6. Under Additional Properties, click **Subscriptions**

Results

The list of subscriptions for the selected topic space is displayed in the content pane. For more information about the list of subscriptions, see “Subscriptions [Collection]” on page 2167.

Each subscription has a unique subscription identifier that has the form, *clientID##subName* where:

clientID
The client identifier used to associate a connection and its objects with the messages maintained for applications (as clients of the JMS provider). You should use a naming convention that helps

you identify the applications, in case you have to relate subscriptions to the associated applications for runtime administration. For more information about client identifiers, see section 4.3.2 of the JMS 1.1 specification.

subName
The JMS subscription name used to uniquely identify a durable subscription within a given client identifier. For more information about JMS subscription names, see section 6.11.1 of the JMS 1.1 specification.

What to do next

To display details of a subscription, click the name of the subscription in the list. To delete one or more subscriptions, select the check box next to the subscription names then click **Delete**.

Stopping active subscribers for durable subscriptions

Use this task to enable applications to change a cloned durable subscription. This task enables applications to connect to an existing cloned durable subscription, and to specify parameters that differ from those that were used to create the existing subscription.

About this task

When an application connects to an existing durable subscription, but specifies parameters that differ from those that were used to create the existing subscription, **the subscription is deleted then recreated with the new parameters**. A durable subscription can be changed in this way only when it has no active consumers.

Note: The *server_name*-durableSubscriptions.ser file in the *WAS_HOME*/temp directory is used by the messaging service to keep track of durable subscriptions for message-driven beans. If you uninstall an application that contains a message-driven bean, this file is used to unsubscribe the durable subscription. If you have to delete the *WAS_HOME*/temp directory or other files in it, ensure that you preserve this file.

To stop active subscribers for one or more durable subscriptions, complete the following steps:

Procedure

1. Use the administrative console to list the durable subscriptions
2. From the list, identify the client identifier of the durable subscription. The name column lists the unique subscription name for each durable subscription, in the form *clientID##subName* where:

   clientID
   The client identifier used to associate a connection and its objects with the messages maintained for the client by the JMS provider.

   subName
   The name used to uniquely identify a durable subscription within a given client identifier.
3. Use your client identifier naming convention to identify the application assigned to the client identifier.
4. List the applications that have active consumers for the durable subscription. In the navigation pane of the administrative console, click **Applications -> Application Types -> WebSphere enterprise applications**.
5. In the console pane, select the check box next to the name of each application that you want to stop.
6. Click **Stop**

Results

This stops the active consumers created by the applications, so applications can reconnect to the durable subscriptions with different parameters from those that were used to create the previous subscriptions.

Deleting durable subscriptions
Use this task to delete a durable subscription.

About this task

To delete one or more durable subscriptions, complete the following steps:

Procedure
1. Display the durable subscriptions
2. Select the check box for each subscription you want to delete.
3. Click **Delete**

Results

The selected durable subscriptions are deleted.

Administering mediations

These topics provide information about mediations, which are used to change how messages are handled at destinations on a service integration bus.

Procedure
- ../ae/cjp_learning.dita
- Mediation programming
- Programming mediations
- “Securing mediations”
- “Configuring mediations” on page 2007
- “Configuring mediation points” on page 2018
- “Managing mediations with administrative commands” on page 2020
- “Operating mediations at mediation points” on page 2020
- “Administering messages on mediation points” on page 2023
- “Message properties support for mediations” on page 2430
- “Error handling in mediations” on page 2434
- “Mediation thread pool properties” on page 2435

Securing mediations

Use the following tasks to secure mediations at an operations level. For example, a mediation inherits its identity from a the messaging engine, but you might want to specify an alternative identity for the mediation to use.

Configuring an alternative mediation identity for a mediation handler
Use this task to configure an alternative mediation identity for a mediation handler

About this task

By default, a mediation inherits the identity used by the messaging engine. In some cases, you might want to specify an alternative identity for a mediation handler to use. For example, for a single mediation that sends messages to a destination. To do this, you specify a "run-as" identity for the mediation handler at deployment, and map the mediation handler to an identity other than the default mediation identity by using a role name. Follow the steps below to specify an alternative mediation identity:

Procedure

1. Package your mediation handler as an EAR file.
2. Edit the deployment descriptor file to define the roles. For more information, see Configuring programmatic logins for Java Authentication and Authorization Service.
3. Assign users to the role. For more information, see Mapping users to RunAs roles using an assembly tool and Securing applications during assembly and deployment.
4. Deploy the mediation handler in WebSphere Application Server, and assign users to the RunAs role. For more information, see Assigning users to RunAs roles. You can confirm the mappings of users to roles, add new users and groups, and modify existing information during this step. For more information, see Deploying secured applications.

Example

What to do next

Next, you are ready to authorize mediations to access destinations. For more information, see Administering authorization permissions.

Configuring the bus to access secured mediations

Use this task to ensure that the service integration bus is authorized to access secured mediations.

Before you begin

The mediation is secured by using a Java Platform, Enterprise Edition (Java EE) Connector Architecture authentication alias. For information about creating a Java EE authentication alias, see ../ae/tsec_j2cauthdata.dita.

About this task

To configure the bus to access a secured mediation, you must add the mediation authentication alias for the secured mediation to the properties for the bus:

- If the bus has a Version 6 bus member, you must provide the principal and its associated password.
- If the bus has WebSphere Application Server Version 7.0 or later bus members only, you need only provide the principal.

Procedure

1. Log into the navigation pane.
2. Click **Service integration -> Buses -> *security_value***. The bus security configuration panel is displayed.
3. In the **Mediations authentication alias** field, select the principal for the mediation, and its associated password if required.
4. Click **OK**.
5. Save your changes to the master configuration.

Results

The selected bus is configured to access secured mediations.

What to do next

You can assign security roles to your mediation handlers to protect them from use by unauthorized users. For more information, see Deploying secured applications.

Configuring a bus to run mediations in a multiple security domain environment

Use this task to configure a secured bus so that it can run mediations successfully on bus members in different security domains.

Before you begin

The secured bus must be configured to use a non-global security domain. For more information about securing buses by using multiple security domains, refer to Securing buses.

About this task

If your bus topology has bus members in different security domains, you can configure the bus to allow mediations to run under the server identity. This means that a mediation can run on any server in any domain. You do not have to add a dedicated user ID for each mediation to the user repository, or maintain a mediation authentication alias.

Use the administrative console to configure a secured bus to run mediations successfully as follows:

Procedure

1. In the navigation pane, click **Service integration -> Buses -> *security_value***. The security settings for the selected bus are displayed.
2. Check the option **Use the Server ID when running mediations**.
3. Click **Apply**.
4. Save your changes to the master configuration.

Results

You have configured the bus to run mediations successfully across servers in multiple security domains.

What to do next

You can use the administrative console to control access to the bus by administering users and groups in the bus connector role.

Configuring mediations

Use the following tasks if you want to modify the behavior of a mediation, control which messages are mediated, or influence how messages are processed.

Installing a mediation

You can install an enterprise application (EAR file) containing a mediation on a server.

Before you begin

- You have an EAR file that you can deploy into WebSphere Application Server (base). The EAR file contains a mediation handler enterprise bean. You can use the tooling provided with IBM Rational Application Developer to assemble EAR files.
- For guidance on selecting the target application server for your mediation, see Mediation application installation.

About this task

In this task, you use the administrative console to install a mediation application into WebSphere Application Server (base). For additional information about installing enterprise applications, see Installing enterprise application files with the console. For information about installing a secure mediation handler, see Deploying secured applications.

Procedure

1. Click **Applications -> New Application -> New Enterprise Application**.
2. On the first Preparing for application install page, specify the full path name of your EAR file, and click **Next**.
3. On the second Preparing for application install page, choose to generate default bindings.
   a. Expand **Choose to generate default bindings and mappings**.
   b. Select **Generate default bindings**. Using the default bindings causes any incomplete bindings in the application to be completed with default values. The product does not change existing bindings.
   c. Click **Next**. The Select Installation options page is displayed.
4. Type the name of your EAR file in the **Application name** field, if it not already specified, and click **Next**. The Map modules to servers page is displayed.
5. Select the target application server where you want to install the modules that are contained in your application, and click **Next**.
6. Optional: If the panel **Deploy enterprise beans** is displayed, click**Apply**, and click **Next**.
7. Review the summary of the installation options you have chosen:
   a. If you want to make changes, click **Previous** to return to the appropriate page.
   b. When you are ready, click **Finish** to confirm your installation options.
8. Save your changes to the master configuration.

Results

Your mediation application is installed on your chosen server or cluster.

What to do next

You can configure the properties for the mediation in the administrative console. For more information, see “Configuring a new mediation.”

Configuring a new mediation

Use this task to configure a mediation in the administrative console.

Before you begin

This task assumes that you have installed a mediation application on the server where you want to run the mediation. For more information, see “Installing a mediation” on page 2007.

About this task

Configuring a new mediation makes it available for use in mediating one or more destinations.

In this task, you use the administrative console to configure a mediation:

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations**.
2. In the content pane, click **New**.
3. Specify the following properties for the mediation:

   Mediation name
   Type a name for the mediation that is unique to the service integration bus. This name is used to identify the mediation for administrative purposes.

   Description
   Optionally, type a description for the mediation.

Handler list name
Type the handler list name. This is the name that was used when the mediation handler was deployed. Handler list names are unique within the WebSphere Application Server administrative cell.

Global transaction
Optionally, select this option (setting it to `true`) if you want to start a global transaction for each message mediated by the mediation. By default, Global transaction is set to false, and a global transaction is not started when a message is mediated.

Allow concurrent mediation
Optionally, select this option (setting it to `true`) if you want to mediate multiple messages at the mediated destination.

By default, Allow concurrent mediation is set to false, and concurrent mediations are not allowed.

Note: Do not allow concurrent mediation if message ordering is important.

Selector
Optionally, type a message selector to control which messages are mediated by the mediation. The selector operates on the content of the message header. The syntax of the message selector is defined by the JMS specification.

If the message content meets the conditions of the selector, the message is mediated. Otherwise, the message is not mediated.

Discriminator
Optionally, type a message discriminator to control which messages are mediated by the mediation. Message discriminators conform to the publish/subscribe topicspace syntax.

If the message meets the conditions of the discriminator, the message is mediated. Otherwise, the message is not mediated.

4. Click **OK**.
5. Save your changes to the master configuration.

Results

You have configured a new mediation.

What to do next

You can use the mediation to mediate a destination. For more information, see “Mediating a destination” on page 2017.

Deleting a mediation

Use this task to delete a selected mediation in the administrative console.

Before you begin

Before deleting a mediation, you must ensure that it is not in use at a bus destination. You cannot delete a mediation that is in use at a destination; you must first unmediate the destination. For more information, see “Unmediating a destination” on page 2017.

About this task

Deleting a mediation removes it from the administrative console. To delete a mediation, use the administrative console to complete the following steps

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations**.
2. Select the mediations that you want to delete.
3. Click **Delete**.
4. Save your changes to the master configuration.

Results

The mediations are deleted.

Modifying the properties of a mediation

Use this task to set properties to control some aspects of the runtime behavior of an existing mediation.

Before you begin

You should ensure that the mediation exists. If you want to configure a new mediation, see "Configuring a new mediation" on page 2008.

About this task

Using the administrative console, you can specify the following options for a selected mediation:

- Each message processed by the mediation runs within a global transaction.
- The mediation processes messages concurrently.
- The mediation processes only those messages that meet some selection criteria.

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name***.
2. Select any of the following properties for the mediation:

   Description
   Optionally, type a description of the behavior of the mediation, taking into account any properties specified for the mediation.

   Handler list name
   Type the name of the handler list that was defined when the mediation was deployed. Handler list names start with an uppercase letter, and are unique within the service integration bus.

   Global transaction
   Optionally, select this option (setting it to `true`) if you want to start a global transaction for each message mediated by the mediation.

   By default, Global transaction is set to false, and a global transaction is not started when a message is mediated.

   Allow concurrent mediation
   Optionally, select this option (setting it to `true`) if you want to mediate multiple messages at the mediated destination.

   By default, Allow concurrent mediation is set to false, and concurrent mediations are not allowed.

   Note: Do not allow concurrent mediation if message ordering is important.

Selector

Optionally, type a message selector to control which messages are mediated by the mediation. The selector operates on the content of the message header. The syntax of the message selector is defined by the JMS specification.

If the message meets the conditions of the selector, the message is mediated. Otherwise, the message is not mediated.

For information about the properties that can be used in selectors, see "Message properties support for mediations" on page 2430.

Discriminator

Optionally, type a message discriminator to control which messages are mediated by the mediation. Message discriminators conform to the publish/subscribe topic space syntax.

If the message meets the conditions of the discriminator, the message is mediated. Otherwise, the message is not mediated.

3. Specify any additional properties.
4. Click **OK**.
5. Save your changes to the master configuration.

Results

You have changed the properties for the selected mediation.

What to do next

- If you want to configure the mediation context information, see "Configuring mediation context properties" on page 2013.
- To use the mediation to mediate a destination, see "Mediating a destination" on page 2017.

Adding mediation context information

Use this task to add a mediation context property for a selected mediation.

Before you begin

The mediation for which you want to add context information must exist in the administrative console. To create a mediation, see "Configuring a new mediation" on page 2008.

About this task

Mediation context information is used, in addition to the message header information, to determine how a message is processed at run time. New context information is defined by adding a mediation context property. A destination can also have context properties. If you add a context property for a mediation that has the same name as a destination context property, the property on the destination takes precedence.

You can add the following mediation context properties:

Property name	Description	Value	Additional information
sib:SkipWellFormedCheck	Omits the well formed check that is performed on messages after they have been processed by the mediation.	True or false	This property is overridden by the message delivery option assured persistent. A well formed check is always performed for messages that have the delivery option assured persistent.

Property name	Description	Value	Additional information
sib:GlobalTransactionLPSEnabled	Allows a mediation to contain resources for which Last Participant Support (LPS) is enabled. This enables a one-phase commit resource to participate in a global transaction that has two-phase commit resources.	True or false	

To add new mediation context information for a selected mediation, use the administrative console to complete the following steps:

Procedure

1. Display the Context Properties panel for a selected mediation: click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name* -> [Additional Properties] Context properties**. Any mediation context properties that have already been defined for the selected mediation are displayed.
2. Click **New** in the content pane.
3. Specify the following information for the mediation context property:

   Name Type a name for the mediation context property. You must type the name exactly as it appears in the table above.

   Context type
   Select the type `Boolean` for the mediation context property.

   Context value
   Type `True` to set the property. The value is not case sensitive.
4. Click **OK**.
5. Save your changes to the master configuration.

Results

You have added new mediation context information for the selected mediation.

What to do next

You can repeat this task to add another mediation context property.

Listing mediation context properties

Use this task to list context properties for a selected mediation.

About this task

Mediation context information is used, in addition to the message header information, to determine how a message is processed at run time. The context information is provided by one or more of the following mediation context properties:

Property name	Description	Value	Additional information
sib:SkipWellFormedCheck	Omits the well formed check that is performed on messages after they have been processed by the mediation.	True or false	This property is overridden by the message delivery option assured persistent. A well formed check is always performed for messages that have the delivery option assured persistent.

Property name	Description	Value	Additional information
sib:GlobalTransactionLPSEnabled	Allows a mediation to contain resources for which Last Participant Support (LPS) is enabled. This enables a one-phase commit resource to participate in a global transaction that has two-phase commit resources.	True or false	

In this task, you use the administrative console to list the mediation context properties for a selected mediation, as follows:

Procedure

1. Log in to the administrative console.
2. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name* -> [Additional Properties] Context properties**.

Results

The mediation context properties for the selected mediation are displayed.

What to do next

You can configure or delete the context information, or add a new context property for the selected mediation.

Configuring mediation context properties

Use this task to configure context properties for a selected mediation.

Before you begin

The mediation context property you want to configure must exist in the administrative console. To add a new mediation context property, see “Configuring a new mediation” on page 2008.

About this task

Mediation context information is used, in addition to the message header information, to determine how a message is processed at run time. By configuring the mediation context properties, you ensure that messages are processed correctly. A destination can also have context properties. If a context property is configured for both a destination and a mediation, with the same name for the property, the property on the destination takes precedence.

You can configure the following mediation context properties:

Property name	Description	Value	Additional information
sib:SkipWellFormedCheck	Omits the well formed check that is performed on messages after they have been processed by the mediation.	True or false	This property is overridden by the message delivery option assured persistent. A well formed check is always performed for messages that have the delivery option assured persistent.

Property name	Description	Value	Additional information
sib:GlobalTransactionLPSEnabled	Allows a mediation to contain resources for which Last Participant Support (LPS) is enabled. This enables a one-phase commit resource to participate in a global transaction that has two-phase commit resources.	True or false	

To configure an existing mediation context property, use the administrative console to complete the following steps:

Procedure

1. Display the mediation context property for a selected mediation: click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name* -> [Additional Properties] Context properties**. The mediation context properties for the selected mediation are displayed.
2. Select the mediation context property you want to configure.
3. Specify the following information for the mediation context property:

   Name Type a name for the mediation context property. You must type the name exactly as it appears in the table above.

   Context type
   Select the type `Boolean` for the mediation context property.

   Context value
   Type a value for the mediation context property: either `True` to set the property, or `False` to unset it. The value is not case sensitive.
4. Click **OK**.
5. Save your changes to the master configuration.

Results

The context property is configured for the selected mediation.

What to do next

You can repeat this task to configure additional mediation context properties.

Deleting mediation context information

Use this task to delete a mediation context property for a selected mediation.

About this task

In this task, you delete a mediation context property. The mediation context information provided by the property is no longer used at runtime to process messages.

You can delete the following mediation context properties:

Property name	Description	Value	Additional information
sib:SkipWellFormedCheck	Omits the well formed check that is performed on messages after they have been processed by the mediation.	True or false	This property is overridden by the message delivery option assured persistent. A well formed check is always performed for messages that have the delivery option assured persistent.

Property name	Description	Value	Additional information
sib:GlobalTransactionLPSEnabled	Allows a mediation to contain resources for which Last Participant Support (LPS) is enabled. This enables a one-phase commit resource to participate in a global transaction that has two-phase commit resources.	True or false	

To delete mediation context information for a selected mediation, use the administrative console to complete the following steps:

Procedure

1. Display the Context Properties panel for a selected mediation: click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name* -> [Additional Properties] Context properties**. The mediation context properties for the selected mediation are displayed.
2. Select the property you want to delete, and click **Delete**.
3. Click **OK**.
4. Save your changes to the master configuration.

Results

Your chosen mediation context property is deleted for the selected mediation.

What to do next

You can repeat this task to delete another mediation context property.

Configuring the mediation thread pool

Use this task to configure the mediation thread pool.

Before you begin

The wsadmin tool must be running. For more information, see Starting the wsadmin scripting client using wsadmin scripting.

About this task

You configure the mediation thread pool if you want to change the number of threads used when running mediations concurrently. The maximum size of the thread pool determines the maximum number of messages that can be mediated concurrently for a messaging engine.

The mediations thread pool, attribute name `mediationsThreadPool`, is an attribute of the messaging engine. By default, `mediationsThreadPool` does not exist, and a default thread pool is created and used at run time. In this task, use the wsadmin tool to create a thread pool object, and then modify its properties using JACL, as shown in the examples below:

Note: You use the wsadmin tool from within Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Procedure

1. Use this example to create a mediationThreadPool object:

```
AdminConfig.create("ThreadPool" , messagingEngine,
                   [["name" , "stitch.server1-bus2-mediationThreadPool"]] ,
                   "mediationThreadPool")
```

Tip: In this case, the thread pool name is based on the name of the messaging engine. Although it is not required to do this, it makes it easier to find the thread pool name when using Performance Monitoring Infrastructure (PMI).

2. Use this example to modify a mediationThreadPool object:

```
AdminConfig.modify(AdminConfig.showAttribute(messagingEngine,
  "mediationThreadPool"), [["maximumSize" , "10"]])
```

`maximumSize` can contain any of the mediationsThreadPool properties. To add additional parameters, insert [*attribute_name attribute_value*] within the outer brackets ([]).

Tip: There is a space between *attribute_name* and *attribute_value*.

Setting tuning properties for a mediation

Use this task to tune a mediation for performance by using the administrative console.

Before you begin

Review the guidance on when it is appropriate to tune a mediation for performance in the topic Performance tuning for mediations.

About this task

You can set the following tuning property in the administrative console to improve the performance of a mediation:

sib:SkipWellFormedCheck
Whether you want to omit the well formed check that is performed on messages after they have been processed by the mediation. Either true or false.

Note: This property is overridden for messages that have the delivery option assured persistent, and a well formed check is always performed.

To set, or unset, one or more tuning properties for a mediation, use the administrative console to complete the following steps:

Procedure

1. Display the mediation context information:
   a. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations**.
   b. In the content pane, select the name of the mediation for which you want to configure tuning information.
   c. Click **[Additional Properties] Context information**.
2. In the content pane, click **New**.
3. Type the name of the property in the **Name** field.
4. Select the type `Boolean` from the drop-down list.
5. Type **true** in the **Context Value** field to set the property, or type **false** to unset the property.
6. Click **OK**.
7. Save your changes to the master configuration.

Mediating a destination

Use the administrative console to mediate a service integration destination with a mediation hosted by service integration. A mediation point is created for the mediated destination on the bus member where you want the mediation to run, and is used to process messages at runtime.

Before you begin

Ensure that the following resources exist:

- The mediation you want to apply to the destination. For more information, see “Configuring a new mediation” on page 2008.
- The bus member where the mediation point is to be created.

About this task

You can mediate a destination with a single mediation only, but you can associate the same mediation with more than one destination. This scenario mediates a destination with a service integration mediation point, and a service integration mediation. If you want to mediate a destination by using an WebSphere MQ program, or assign it to an WebSphere MQ queue point, see “Mediating a destination by using a WebSphere MQ queue as the mediation point” on page 581.

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations**. The bus destinations for the selected bus are displayed.
2. Select the destination you want to mediate, and click **Mediate**. A list of available mediations is displayed.
3. Select the mediation to associate with the destination, and click **Next**.
4. Select the bus member where you want the mediation to run, and click **Next**.
5. Click **Finish** to mediate the destination, or **Previous** to make any changes to your selections.
6. Save your changes to the master configuration.

Results

Your chosen destination is mediated with your chosen mediation, and a mediation point is created for the destination on the bus member.

What to do next

You can control some aspects of how the mediation processes messages at the mediation point. For example, whether the mediation run within a global transaction, or processes multiple messages concurrently. For more information, see “Modifying the properties of a mediation” on page 2010.

Unmediating a destination

Use this task to unmediate a selected bus destination.

Before you begin

Before you start this task, you should consider whether you want to preserve message ordering at the bus destination you intend to unmediate.

About this task

Unmediating a selected bus destination breaks the association between the bus destination and the mediation, and the mediation stops processing messages at the bus destination at run time. This task unmediates a bus destination, and optionally preserves message ordering at the bus destination.

To unmediate a bus destination use the administrative console to complete the following steps:

Tip: If you want to preserve message ordering, complete all the following steps. If message ordering is not important, you can omit steps 2 and 4:

Procedure

1. Display the bus destination you want to unmediate. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
2. Optional: If you want to preserve message ordering at the destination, take the following steps:
   a. Clear the **Send Allowed** check box (setting it to false) to stop sending messages to the message points for this destination.
   b. Wait until there are no messages listed for the mediation points.
   c. Stop all the mediation points, and wait for all the mediation points to enter the `stopped` state.
   d. Optional: If it is not already selected, select the bus destination you want to unmediate.
3. Click **Unmediate** to unmediate the destination.
4. Optional: Select the **Send Allowed** check box (setting it to `true`) to resume sending messages to the mediation points for this destination.
5. Save your changes to the master configuration.

Results

The destination is unmediated, and the mediation point is deleted from the destination.

If you did not preserve message ordering, any messages that remain on the mediation point after you have unmediated the destination are processed as for a non-mediated destination. Messages with an empty forward routing path are placed on the next appropriate message point for the destination. Messages with a non-empty forward routing path are sent to the destination specified in the forward routing path.

The unmediated destination now operates as a non-mediated destination, and new messages are sent to a message point on the destination.

Configuring mediation points

In service integration technologies, messages are held at specialized message points called mediation points before they are processed by a mediation. Use the following tasks to set properties at mediation points to route messages and start and stop mediations:

Configuring a mediation point

You can use the administrative console to browse or change the configuration properties for a selected mediation point for a selected bus destination.

Before you begin

You must first create a mediation point on a bus destination. For more information, see “Mediating a destination” on page 2017.

About this task

A mediation point is a specialized message point on a bus location where messages are stored and mediated. Configuring the properties of a mediation point enables you to control some aspects of how messages are sent to and received from a bus destination. In this task you use the administrative console to configure the properties for a selected mediation point for a selected bus destination.

Note: You should be aware that the mediation point properties Send allowed and Initial state override the general properties defined for the bus destination.

To browse or change the properties for a selected mediation point, use the administrative console to complete the following steps:

Procedure

1. Start the administrative console, and click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points -> *mediation_point_name***. The properties for the selected mediation point are displayed.
2. Optional: Browse or change the following properties for the mediation point:

   Identifier
   The identifier by which this message point is known for administrative purposes. This field is for display purposes only.

   UUID The universal unique identifier assigned to this mediation point for administrative purposes. This field is for display purposes only.

   High message threshold
   The maximum total number of messages that can be placed on this mediation point. This field is for display purposes only.

   Send allowed
   This property specifies whether messages are sent to this mediation point. By default, `True` is selected, and messages are sent to this mediation point. Select `False` to prevent messages from being sent to this mediation point.

   Initial state
   This property specifies the state of the mediation point when the host messaging engine is started for the first time. Permitted values are `Started` or `Stopped`. The default state is `Started`. The mediation does not start mediating messages until Initial state is `Started`.

   Target UUID
   The UUID of the bus destination where this mediation point is created. This field is for display purposes only.
3. Click **OK**.
4. Save your changes to the master configuration.

Results

You have browsed or changed the configuation properties for the selected mediation point.

Listing mediation points for a bus destination

Use this task to list mediation points for a selected bus destination.

About this task

This task lists the mediation points for a selected bus destination. You might want to do this, for example, to check that a destination is mediated.

To display a list of mediation points for a bus destination, use the administrative console to complete the following steps:

Procedure

1. Click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name***.
2. Click **[Message Points] Mediation Points**.

Results

All the mediation points for the selected bus destination are displayed.

Listing mediation points for a messaging engine

You can use the administrative console to list all the known mediation points for a selected messaging engine. You can then view the properties for a mediation point, administer messages for a mediation point, or operate mediations at a mediation point.

About this task

To list the mediation points for a selected messaging engine, use the administrative console to complete the following steps:

Procedure

Start the administrative console, and click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points**.

Results

The Mediation points panel displays a list of all the mediation points for the selected messaging engine.

What to do next

You can click the name of a mediation point in the list to work with its properties, administer its messages or operate mediations.

Managing mediations with administrative commands

You can use wsadmin commands to administer service integration technologies mediations. For example you can create, delete and view mediations, configure mediation properties and mediate destinations.

About this task

These commands provide an alternative to using the administrative console.

Procedure

1. Open a wsadmin command session in local mode For example:

   ```
   wsadmin -conntype none -lang jython
   ```

   Note: The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

2. Type `AdminTask.`*`command`* where *command* is the command format as indicated in the related reference topics. For example:

   ```
   wsadmin>AdminTask.listSIBMediations("-bus abus")
   (cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098217858584)
   (cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098220051588)
   ```

Operating mediations at mediation points

Use these tasks to start, stop and restart mediations at runtime at specialized message points called mediation points.

Before you begin

These tasks assume that mediations are configured in the administrative console. For more information, see “Configuring mediations” on page 2007.

Starting a mediation

Use this task when you want a mediation to apply its message processing to messages at a selected specialized message point on a destination, at runtime.

Before you begin

The mediation you want to start must be associated with a destination. For more information, see “Mediating a destination” on page 2017.

About this task

Starting (and stopping) a mediation is a runtime operation. All mediation runtime operations are performed on specialized destination message points called mediation points. When you start a mediation, the mediation begins operating on messages at a selected mediation point. You can start a mediation after the server has started. A mediation cannot operate on new messages at a mediation point after server shutdown begins.

Tip: To ensure that a mediation operates on a specific message at a mediation point, remove all existing messages from the mediation point before you start the mediation. For more information, see “Deleting messages on a mediation point” on page 2023.

To start a mediation, complete the following steps in the administrative console:

Procedure

1. List the mediation points by using the administrative console:
   - To list the mediation points for a bus destination, click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name* -> [Message Points] Mediation points**.
   - To list the mediation points for a messaging engine, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points**.
2. Select the mediation name.
3. Click **Start**.

Results

The mediation starts operating on messages at the selected mediation point.

What to do next

To list or delete messages at selected mediation points, see “Administering messages on mediation points” on page 2023. To restart a mediation, refer to “Restarting a mediation that has stopped on error” on page 2022.

Stopping a mediation

Use this task to stop a selected mediation at a mediation point.

Before you begin

Before you can operate a mediation, it must be associated with a destination. For more information, see “Mediating a destination” on page 2017.

About this task

Stopping a mediation, as with all runtime operations on mediations, is carried out at a specialized message point on the destination, called a mediation point. By stopping a mediation, you stop any further messages from being delivered to the mediation point. Messages sent to the destination are held at the pre-mediated part of the destination. The messages are not mediated and are not made available to message consumers until the mediation is restarted. Note that mediations starts operating on messages after the server has started and no new messages are mediated after server shutdown begins.

To stop a selected mediation, use the administrative console to complete the following steps:

Procedure

1. List the mediation points.
   - To list the mediation points for a bus destination, click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name* -> [Message Points] Mediation points**.
   - To list the mediation points for a messaging engine, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points**.
2. Select the mediation name.
3. Click **Stop**.

Results

The mediation is stopped at the mediation point. Messages arriving at the mediation point are not processed by the mediation. The messages are stored at the mediation point until the mediation is restarted.

Restarting a mediation that has stopped on error

You must restart a mediation that has stopped on error so that the mediation can continue processing messages. When the mediation is restarted, the mediation resumes mediating messages on the pre-mediated part of the destination.

About this task

Changes in the state of a mediation usually occur as the result of operations performed by the administrator. If the state of a mediation moves from `Started` to `Stopped on error`, and you have not stopped it intentionally, you should investigate the cause of the problem, and resolve it before attempting to restart the mediation.

To restart a mediation that has stopped on error, use the administrative console to complete the following steps:

Procedure

1. List the mediation points.
   - To list the mediation points for a bus destination, click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name* -> [Message Points] Mediation points**.

- To list the mediation points for a messaging engine, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points**.

2. In the content pane, select the mediation name.
3. Click the **Runtime** tab.
4. Under **General Properties**, view the information in the **Reason** field.
5. Resolve the problem.
6. Restart the mediation so that it can continue processing messages.

Administering messages on mediation points

Use these tasks to list and delete runtime messages held at mediation points in a service integration bus.

Listing messages at a mediation point

Use this topic to list messages at a mediation point.

About this task

The mediation point is a specialized message point that defines the association between a mediation and a destination. You can browse and delete messages on a mediation point for a selected bus destination or messaging engine.

To display a list of messages on a selected mediation point, use the administrative console to complete the following steps:

Procedure

1. List the mediation points.
   - To list the mediation points for a bus destination, click **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name* -> [Message Points] Mediation points**.
   - To list the mediation points for a messaging engine, click **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points**.
2. Click the mediation point name. This displays the properties of the mediation point.
3. Click the **Runtime** tab.
4. Click **[Additional Properties] Messages**.

Results

A list of messages on the selected mediation point is displayed in the content pane.

Deleting messages on a mediation point

Use this topic to delete messages that exist on a mediation point for a selected bus destination.

About this task

The mediation point is a specialized message point that defines the association between a mediation and a destination. As with other message points, you can delete messages on a mediation point for a selected bus destination or messaging engine. For example, if you want to preserve message ordering at a destination before it is unmediated, you can delete all the messages at that destination.

Procedure

1. List the messages at the mediation point. For more information, see “Listing messages at a mediation point.”

2. In the content pane, select the check box next to each message you want to delete. Alternatively, select all the messages in the list by clicking **Select All Items**.
3. Click **Delete**.

Results

The selected messages are removed from the destination.

Example: Using mediations to trace, monitor and log messages

The most straightforward use of a mediation is for tracing, monitoring or logging messages that pass through a destination or topics spaces. This type of mediation does not modify the message; it only extracts information from the message and saves or displays the information elsewhere.

For example, the following mediation handler displays the API message and correlation IDs for each message it is sent:

```
public boolean(MessageContext context)
{
  SIMessageContext   msgCtx  = (SIMessageContext)context;
  SIMediationSession session = msgCtx.getSession();
  SIMessage          msg     = msgCtx.getMessage();
  String             msgId   = msg.getApiMessageId();
  String             corrId  = msg.getCorrelationId();
  String             dest    = session.getDestinationName();

  System.out.println(msgId+" (correlation id="+corrid) is passing through "+dest+".");

  return true;
}
```

SIB service [Settings]

The service that provides service integration functions.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] SIB service.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Enable service at server startup:

Specifies whether or not the server attempts to start the specified service when the server starts.

If you want applications to use service integration technologies provided by this application server, including JMS resources of the default messaging provider of this application server, this option must be selected. The SIB Service is enabled automatically when you add a server to a service integration bus. You can choose to disable the SIB service, for example if you have removed the only bus member for a server.

For information about using service integration technologies, see Administering service integration.

Required	No
Data type	Boolean

Configuration reload enabled:

Select this option to enable the dynamic reloading of the SIB configuration files for this server.

The information that defines the configuration of service integration buses and their resources is saved in a set of configuration files. When a server starts up, it uses the current information about service integration read from those configuration files. When a messaging engine is started, it uses the information in the server that it is running in.

If the information in the configuration files is changed while the server is running, the server must either be dynamically updated or restarted to use the updated information.

With dynamic reloading of configuration files, any updates to the configuration information are dynamically passed to the server, and therefore made available to messaging engines whether or not they are started. You can enable dynamic reloading of configuration files for servers and for service integration buses.

If you choose not to enable dynamic reloading of configuration files, you must restart the server to pick up any changes to the configuration files.

In a cluster deployment with failover, the configuration information is likely to be updated between the initialization and start up of a messaging engine (A messaging engine is initialized when the server starts, but might not get started for a long time after that). Therefore you should enable dynamic reloading of configuration files in a cluster deployment with failover, because restarting a server to pick up configuration changes causes a failover. To get predictable behavior on failover, you must ensure that the standby (inactive) servers had been updated and recycled before the active server.

Required	No
Data type	Boolean

Add a transport to the list of permitted transports [Settings]

Add a transport to the list of permitted transports.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Additional Properties] Permitted transports -> *member_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Transport chain name:

Select the transport name from the list of chain names.

You can manage inbound chains through the administrative console by selecting either **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine**

inbound transports or **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] WebSphere MQ link inbound transports**. You can also use these administrative console panels to define new transport chains from a set of templates.

The outbound chains that an application server uses for bootstrap operations are defined when the server is defined. They can be altered, or new bootstrap chains can be defined, by using the wsadmin tool.

Required
No

Data type
drop-down list

Range

InboundBasicMessaging
This transport chain allows communication through the TCP protocol. The default port used by this chain for the first server on the node is 7276. You should verify that the selected port is not already used, for example if you are configuring a second server with the same name as the first server. Messaging engines hosted in other application servers and JMS applications running in a client container can communicate with the messaging engines of the server by using this transport chain.

InboundSecureMessaging
This transport chain provides secure communication by using the secure sockets layer (SSL) based encryption protocol over a TCP network. The default port used by this chain for the first server on the node is 7286. You should verify that the selected port is not already used, for example if you are configuring a second server with the same name as the first server. The SSL configuration information for this chain is based on the default SSL repertoire for the application server. Messaging engines hosted in other application servers and JMS applications running in the client container can communicate using this transport chain.

InboundBasicMQLink
This transport chain supports WebSphere MQ queue manager sender channels and applications by using the WebSphere MQ JMS provider connecting over a TCP network. The default port used by this chain is 5558, although this can be automatically adjusted to avoid conflicts.

InboundSecureMQLink
This transport chain enables WebSphere MQ queue manager sender channels and applications by using the WebSphere MQ JMS provider to establish SSL based encrypted connections over a TCP network. The default port used by this chain is 5578, although this is automatically adjusted to avoid conflicts.

Range (continued)

BootstrapBasicMessaging
This transport chain is used to establish bootstrap connections to inbound chains configured for TCP-only connections to an application server, such as the InboundBasicMessaging chain.

BootstrapSecureMessaging
This transport chain is used to establish secure connections by using secure sockets layer (SSL) based encryption. The SSL configuration used is taken from the default SSL repertoire when used in an application server environment or from a configuration file when used by the client container. This chain can be used for establishing bootstrap connections to inbound chains that are configured to use SSL, for example, the InboundSecureMessaging chain. Success in establishing such a connection depends on a compatible set of SSL credentials being associated with both this bootstrap outbound chain and also with the inbound chain to which the connection is being made.

BootstrapTunneledMessaging
This transport chain is used to establish bootstrapping connections that are tunneled through HTTP. This transport chain tunnels JFAP and uses HTTP wrappers. Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.)

BootstrapTunneledSecureMessaging
This transport chain tunnels JFAP and uses HTTP wrappers. Used to establish bootstrapping connections that are tunneled through secure HTTP (HTTPS). Like the BootstrapSecureMessaging outbound chain, this chain also derives its SSL configuration from the default SSL repertoire when used in an application server or from a configuration file when used in the client container.

Range (continued)

OutboundBasicMQLink
This transport chain is used to establish connections with WebSphere MQ queue manager receiver channels. It is used when communicating with WebSphere MQ through a WebSphere MQ link.

OutboundSecureMQLink
This transport chain is used to establish connections with WebSphere MQ queue manager receiver channels that have been secured by using SSL. It is used when communicating with WebSphere MQ through a WebSphere MQ link. The SSL configuration used is taken from the default SSL repertoire for the application server being used to contact the queue manager.

OutboundBasicWMQClient
This transport chain is used when communicating with WebSphere MQ receiver channels. It is used when communicating with WebSphere MQ through a WebSphere MQ server.

OutboundSecureWMQClient
This transport chain is used when communicating with WebSphere MQ receiver channels that have been secured by using SSL. It is used when communicating with WebSphere MQ through a WebSphere MQ server.

Alias destination [Settings]

An alias destination makes a destination available by another name and, optionally, overrides the parameters of the destination.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The identifier by which this destination is known for administrative purposes.

The identifier and bus are used to identify this destination to applications.

Required	No
Data type	String

Bus:

The name of the bus on which this destination is known to applications.

The identifier and bus are used to identify this destination to applications.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this destination for administrative purposes.

Required	No
Data type	String

Type:

Whether this bus destination is for a queue, topic space, or some other type of destination.

Required	No
Data type	String

Description:

An optional description for the bus destination, for administrative purposes.

Required	No
Data type	Text area

Target identifier:

The name of the destination for which this is an alias.

Note: The targetIdentifier field is read-only and cannot be modified; if you want to change the target of an alias, you must delete the alias destination then re-create it.

Required	No
Data type	String

Target bus:

The name of the bus on which the destination for which this is an alias exists.

If you do not specify a bus name, the target destination is assumed to be on the same bus as the alias destination.

Note: The targetBus field is read-only and cannot be modified; if you want to change the target of an alias, you must delete the alias destination then re-create it.

Required	No
Data type	String

Enable producers to override default reliability:

Select this option to enable producers to override the default reliability that is set on the destination.

- If an application is producing messages to a destination and that application does NOT specify a reliability for the message, then the Default reliability setting is used.
- If an application is producing messages to a destination and that application specifies a reliability, then the application reliability is ONLY applicable if the Enable producers to override default reliability property is set to True. Otherwise, the Default reliability setting is used.

Required No
Data type drop-down list
Range

Inherit The alias destination uses (inherits) the value of the corresponding property on the target destination.

True The alias destination uses the delivery option value specified by producers.

False The alias destination uses the delivery option value specified by the Reliability property of the destination.

Default reliability:

The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

This reliability setting is used if a message has not set a reliability or if the “Enable producers to override default reliability” is set to False.

- If an application is producing messages to a destination and that application does NOT specify a reliability for the message, then the Default reliability setting is used.
- If an application is producing messages to a destination and that application specifies a reliability, then the application reliability is ONLY applicable if the Enable producers to override default reliability property is set to True. Otherwise, the Default reliability setting is used.

Required No
Data type drop-down list

Range

Inherit The alias destination uses (inherits) the value of the corresponding property on the target destination.

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Maximum reliability:

The maximum reliability of messages accepted by this destination.

Producers cannot send messages to this destination with a reliability higher than the value specified for this property.

Required No
Data type drop-down list

Range

Inherit The alias destination uses (inherits) the value of the corresponding property on the target destination.

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Default priority:

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

Required No
Data type Integer
Range -1 through 9

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

Required No
Data type drop-down list
Range

Inherit The alias destination uses (inherits) the value of the corresponding property on the target destination.

True The alias destination uses the delivery option value specified by producers.

False The alias destination uses the delivery option value specified by the Reliability property of the destination.

Receive allowed:

Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination.

Required No
Data type drop-down list
Range

Inherit The alias destination uses (inherits) the value of the corresponding property on the target destination.

True The alias destination uses the delivery option value specified by producers.

False The alias destination uses the delivery option value specified by the Reliability property of the destination.

Reply destination:

The name of a destination to be appended to any non-empty reverse routing path of messages sent to this destination.

This property is intended for use with mediations on reply messages.

For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

Required No
Data type String

Reply destination bus:

The bus on which the reply destination exists.

This property is intended for use with mediations on reply messages.

For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

Required No
Data type String

Default forward routing path:

The value to which a message's forward routing path will be set if the message contains no forward routing path. This identifies a sequential list of intermediary bus destinations that messages must pass through to reach a target bus destination. The format of the field is a list of line-delimited bus destinations specified as *bus*:*name*.

If you want to forward messages to one or more bus destinations, type a list of bus destinations. Type each destination entry on a separate line, and in the form *bus_name*:*destination_name* or :*destination_name*

Where

bus_name
: Is the name of the service integration bus on which the destination is configured. If you do not specify a bus name, the destination is assumed to be on the same bus as the destination for which you are setting this property.

destination_name
: is the name of a bus destination.

Required No
Data type Text area

Delegate authorization check to target destination:

Indicates whether the authorization check is performed on the alias or the target destination.

Indicates which destination access role is checked when a user accesses the alias destination. When this option is selected, the destination access role of the target destination is checked. When this option is not selected, the destination access role of the alias destination is checked. If you do not want to override the security of the target destination, select this option.

Required No
Data type Boolean

Include an RFH2 message header when sending messages to WebSphere MQ:

If selected, messages sent to WebSphere MQ include an RFH2 header. The RFH2 header stores additional information to that which is stored in theWebSphere MQ message header.

This property applies when the target bus is a WebSphere MQ queue manager or queue-sharing group. When service integration converts a message from the service integration format to WebSphere MQ format, by default it includes an MQRHF2 header in the WebSphere MQ message. This header contains message attributes, such as JMS message attributes, that are not WebSphere MQ message attributes and therefore do not appear in the WebSphere MQ message descriptor (MQMD). Some WebSphere MQ applications cannot process messages that include an MQRFH2 header. If messages sent to this destination will be processed by WebSphere MQ applications that cannot tolerate an MQRFH2, clear this option.

Dynamic updates to this property are effective immediately.

Required No
Data type Boolean

Use all target queue points:

Whether to use all target queue points

Note: This option is only visible when an alias destination is defined to point at a destination with multiple queue points.

Indicates whether all the queue points of the target queue can be used, including any queue points created after the alias is configured. When selected, the **Target queue points** menu is disabled.

Required Yes

Data type	Boolean

Unselected queue points:

This is a list of the queue points that are not addressable by the alias definition. The list is generated from the complete list of queue points for this queue.

Note: This option is only visible when an alias destination is defined to point at a destination with multiple queue points.

This list is enabled only when **Use all target queue points** is unchecked.

Required	No
Data type	drop-down list

Selected queue points:

This is a list of the queue points that are addressable by the alias definition. Messages produced to or consumed from this alias apply to only these queue points.

Note: This option is only visible when an alias destination is defined to point at a destination with multiple queue points.

This list is enabled only when **Use all target queue points** is unchecked.

Required	No
Data type	drop-down list

Additional Properties

Context properties
Context information passed to the mediation.

Related Items

Application resources topology
A expandable tree view of all applications and messaging resources that reference the current destination.

Application resources for this destination

This pane provides an expandable tree view of all the applications and messaging resources that reference the current destination, both directly and indirectly. As many of the references as possible are resolved to links to the associated configuration panel for the referenced object.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Related Items] Application resources topology**
- **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name* -> [Related Items] Application resources topology**

Use this panel to inspect the configuration from the queue or topic space (destination) to the application or other JMS resources to ensure that the configuration is correct.

Using the default messaging provider, a JMS activation specification refers to a JMS destination (either a JMS Queue or JMS Topic) that in turn points to a service integration bus destination (either a queue destination or a topic space destination). JMS resources are referenced through Java Naming and Directory Interface (JNDI) names and service integration resources are referenced through bus and resource identifiers. This panel enables you to review the dependencies within your configuration, making it easier to detect inconsistencies, for example as a result of references being entered incorrectly, or JNDI or resource names being changed or deleted without the associated configuration having been updated.

Problems with the configuration are usually detected in one of two ways:

- An application can no longer send or receive messages.
- A destination becomes full and can no longer receive messages because the existing messages are not being consumed.

This panel can help you find the cause of the problem by giving you a high level view of many relevant resources.

Note: For a related view of the JMS resources for a given application, see the following panel:
"Messaging resources for this application" on page 2129.

- "Local Topology tab"

Local Topology tab

Topology properties for this object. These properties detail how this object relates to other objects in the system topology.

Bootstrap members [Collection]

Bus members, servers and clusters that client applications can target to bootstrap into the given bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Bootstrap members.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This panel displays the bootstrap members for the selected bus. A bootstrap member is an application server or cluster that has been configured to accept bootstrap requests into the bus. The bootstrap member authenticates a connection request, and directs the request to a bus member. Select a bootstrap member policy to restrict which of the following types of server can service requests to bootstrap into the bus:

All members of the cell with the Service Integration Bus Service enabled
This the default policy. Use any server in the cell that has the Service Integration Bus Service enabled to service bootstrap requests.

Bus members and nominated bootstrap members
Use bus members and nominated cell members to service bootstrap requests.

Bus members only
Use bus members only to service bootstrap requests.

Click **Apply** for the policy to take effect.

Name The server or cluster name for the bootstrap member.

Type The type of bootstrap member. Values are Bus member, Nominated member or SIB Service enabled.

SIB Service
Whether SIB Service is enabled on this member. Values are Enabled or Disabled.

Host The host name for the node on which the bootstrap member is located.

Ports The ports from which the server can accept bootstrap requests.

Buttons

New	This button is only displayed if you have applied the bootstrap member policy option “Bus members and nominated bootstrap members”. Click to start the Add bootstrap member wizard.
Delete	Click to remove the selected bootstrap member. This button is only displayed if you have applied the bootstrap member policy option “Bus members and nominated bootstrap members”. You cannot remove bus members, or members for which SIB Service is enabled.

Bus members [Settings]

Bus members are the servers, WebSphere MQ servers and clusters that have been added to the bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Bus members -> *bus_member_name*.

Use this pane to view or change messaging engine settings for a cluster when you are using messaging engine policy assistance.

You can change the currently selected messaging engine policy assistance, view the configuration diagram for the cluster, undertake further messaging engine configuration steps by using the links available in Additional Properties, or disable messaging engine policy assistance.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the bus member.

This field is read-only.

Required No
Data type String

Type:

The resource type of the bus member (Server, Server Cluster, or WebSphere MQ Server).

This field is read-only.

Required	No
Data type	String

Enable messaging engine policy assistance:

A value that shows whether messaging engine policy assistance is set for the server cluster. This option is always selected when you first view this pane. To disable messaging engine policy assistance, deselect this option. The list of messaging engine policy assistance types and the configuration diagram are no longer displayed.

Policy name:

A value that shows the currently selected messaging engine policy assistance type for the server cluster. To change the selection, click the required radio button in the Select column, then click **Apply** or **OK**. For each messaging engine policy assistance type, except Custom, the **Is further configuration required?** column shows `No` if the number of messaging engines and their behavior are correctly configured. Otherwise, this column shows one or more messages with suggested actions to achieve the number of messaging engines and the required messaging engine behavior for each messaging engine policy assistance type. A diagram shows the servers, the currently configured messaging engines, and an indication of messaging engine behavior in the cluster. It also shows any changes required to support the currently selected messaging engine policy assistance type.

Additional Properties

Messaging engines
Use the **Messaging engines** option to view or change the messaging engines in this cluster. For example, you can add, remove, start or stop a messaging engine.

Messaging engine policy maintenance
Use the **Messaging engine policy maintenance** option to view or correct the policy settings for the messaging engines in this cluster, that is, the settings that affect messaging engine behavior.

Redundant core group policies
Use the **Redundant core group policies** option to view or delete any core group policies that are associated with this cluster, but that are not associated with any messaging engine in this cluster. When you use messaging engine policy assistance, you do not have to create or manage core group policies directly. A redundant core group policy might occur if messaging engine policy assistance is not enabled and you remove a messaging engine without deleting its associated core group policy, or create a core group policy that is not associated with a messaging engine.

Bus members [Collection]

Bus members are the servers, WebSphere MQ servers and clusters that have been added to the bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Bus members.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Server clusters can be used as bus members only in WebSphere Application Server environments that support server clusters.

Name The name of the bus member.

Type The resource type of the bus member (Server, Server Cluster, or WebSphere MQ Server)

Messaging engine policy assistance
A value that shows whether messaging engine policy assistance is set for the bus member when the bus member is a server cluster. If the **Type** is Server or WebSphere MQ Server, the value is Not applicable. For a **Type** of Cluster, the value can be Disabled, Enabled (High availability), Enabled (Scalability), Enabled (Scalability with high availability), or Enabled (Custom). The words in parentheses show the type of messaging engine policy assistance that is set. If the current configuration of the cluster is either not suitable or not complete for the type of messaging engine policy assistance that is set, a warning icon is also displayed.

Buttons

Add	Add a server, a cluster, or a WebSphere MQ server to a bus.
Remove	Remove the selected bus members from the bus.

Buses [Collection]

A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

To view this page in the console, click the following path:

Service integration -> Buses.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A messaging engine manages messaging resources and, through destinations assigned to the messaging engine, provides a connection point to which both local and remote applications connect to access messaging resources on the bus. If you add an application server or a server cluster as a bus member, a messaging engine is automatically created for this new member. If you add the same server as a member of multiple buses, the server is associated with multiple messaging engines (one messaging engine for each bus). You can create additional messaging engines for use with server clusters that are bus members, for availability and scalability reasons. However, in its simplest form a bus can be realized by a single engine.

The functions of service integration buses comprise the *SIB service*, which is available on each application server in the WebSphere Application Server environment. By default, the SIB service is disabled, so when a server starts it cannot undertake any messaging. If you add the server to a service integration bus, the SIB service is automatically enabled. If required, you can disable the service again by configuring the server.

The bus appears to its applications as if it were a single logical entity, which means applications only have to connect to the bus and do not have to be aware of the bus topology. In many cases the knowledge of how to connect to the bus and of which bus resources are defined are handled by a suitable API abstraction, such as the administered JMS connection factory and JMS destination objects

Name The name of the service integration bus. Choose a unique name.

Note: The system is unable to differentiate between upper and lowercase characters in bus names. For example, you will not be able to create two buses named `BUS1` and `bus1` because they will not be recognized as different to each other.

Description
An optional description for the bus, for administrative purposes.

Security
Security of your service integration bus can be managed from here.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Buses [Settings]

A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name*.

Server clusters can be used as bus members only in WebSphere Application Server environments that support server clusters.

- “Configuration tab”
- “Local Topology tab” on page 2044

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the service integration bus. Choose a unique name.

Note: The system is unable to differentiate between upper and lowercase characters in bus names. For example, you will not be able to create two buses named `BUS1` and `bus1` because the names will not be recognized as different to each other.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this bus for administrative purposes.

Required	No
Data type	String

Description:

An optional description for the bus, for administrative purposes.

Required	No
Data type	Text area

Inter-engine transport chain:

The transport chain used for communication between messaging engines in this bus.

The transport chain must correspond to one of the transport chains defined in the Messaging engine inbound transports settings for the server. All servers automatically have a number of transport chains defined to them, and it is also possible to create new transport chains.

The default transport chain is InboundBasicMessaging.

For more information see ../ae/cjr0490_.dita.

Required	No
Data type	String

Discard messages:

Whether messages on a deleted message point should be retained at a system exception destination or can be discarded.

Required	No
Data type	Boolean

Configuration reload enabled:

Select this option to enable certain changes to the bus configuration to be applied without requiring the messaging engines to be restarted.

Select this option to enable automatic update of configuration information about all the messaging engines on the bus.

Changes to bus destinations or mediations are applied when destinations or mediations are added to or removed from the bus.

Changes to the modifiable configuration information for any foreign bus connections are also updated automatically. The time when these changes take effect varies:

Foreign Bus Connection properties
Immediately

WebSphere MQ link properties
On channel restart, except Description (immediately), and Initial State (on messaging engine restart)

MQ sender channel properties
On channel restart, except Initial State (on messaging engine restart or sender channel creation)

MQ receiver channel properties
On channel restart, except Initial State (on messaging engine restart or receiver channel creation)

Publish/subscribe broker profile (0 to n) properties
Immediately

Service integration bus link properties
On link restart, except Description (immediately), and Initial State (on messaging engine restart or link creation)

The **Configuration reload enabled** property also needs to be set on the SIB Service of the application server. To ensure that dynamic configuration updates are made on an application server, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] SIB service** to display the Application Servers window, then select **Configuration reload enabled**.

To ensure that dynamic configuration updates are made to each node, click **System administration -> Console Preferences** to display the Console Preferences window then select **Synchronize changes with Nodes**.

Required	No
Data type	Boolean

Default messaging engine high message threshold:

A threshold above which the messaging system will take action to limit the addition of more messages to a message point.

When a messaging engine is created on this bus, the value of this property is used to set the default high message threshold for the messaging engine.

Required	No
Data type	Long
Range	1 through 9223372036854775807

Bootstrap members:

Bus members, servers and clusters that client applications can target to bootstrap into the given bus.

Select one of the following bootstrap member policies to limit the range of available bootstrap members:

All members of the cell with the Service Integration Bus Service enabled
This the default policy. Use any server in the cell that has the Service Integration Bus Service enabled to service bootstrap requests.

Bus members and nominated bootstrap members
Use bus members and nominated cell members to service bootstrap requests.

Bus members only
Use bus members only to service bootstrap requests.

Required	No
Data type	Radio button

Topology

Bus members
New bus member link

Messaging engines
A messaging engine manages bus resources and provides a connection point for applications.

Foreign bus connections
A foreign bus is another bus with which this bus can exchange messages.

Bootstrap members
Bus members, servers and clusters that client applications can target to bootstrap into the given bus.

Destination resources

Destinations
A bus destination is a logical address within a service integration bus.

Mediations
Mediations define the information needed by a messaging engine to perform the mediation processing for associated destinations.

Services

Inbound services
An inbound service describes the web service enablement of a service destination. It provides the configuration of endpoint listeners within a port.

Outbound services
An outbound service represents a WSDL-described service.

WS-Notification services
A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

Reliable messaging state
Use this page to view and manage the WS-ReliableMessaging runtime state.

Additional Properties

Custom properties
Arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

Security
Security of your service integration bus can be managed from here.

Local Topology tab

Topology properties for this object. These properties detail how this object relates to other objects in the system topology.

Context properties [Collection]

Context information used to enable correct processing of messages. This information adds to the context information derived from processing the message header.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Additional Properties] Destination defaults -> [Additional Properties] Context information**
- **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name* -> [Additional Properties] Context properties**
- **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name* -> [Additional Properties] Context information**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Dynamic updates to this list are effective immediately.

Name The name of this context property. The mediation will retrieve the context property using this name.

Context type
The type of the context property, for example, Boolean, Byte, or Character.

Context value
The value of the context property.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Context properties [Settings]

Context information used to enable correct processing of messages. This information adds to the context information derived from processing the message header.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Additional Properties] Destination defaults -> [Additional Properties] Context information -> *context_information_item_name***
- **Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name* -> [Additional Properties] Context properties -> *context_information_item_name***

- **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name* -> [Additional Properties] Context information -> *context_information_item_name***

You can use this panel to view or change context information settings. This panel is useful for configuring extra context information, because it is not possible to anticipate in advance all of the context information that might be required to allow correct processing of messages.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of this context property. The mediation will retrieve the context property using this name.

Required Yes
Data type String

Context type:

The type of the context property, for example, Boolean, Byte, or Character.

Required No
Data type drop-down list
Range

Boolean This context has a Boolean information type.

Byte This context has a byte information type.

Character This context has a character information type.

Double This context has a double information type.

Float This context has a floating point information type.

Integer This context has an integer information type.

Long This context has a long information type.

Short This context has a short information type.

String This context has a string information type.

Context value:

The value of the context property.

Required Yes
Data type String

Custom properties [Collection]

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Additional Properties] Custom properties.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Custom properties [Collection]

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Custom properties.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Custom properties [Settings]

Arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Custom properties -> *property_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

Specifies the name of the property.

Required	Yes
Data type	String

Value:

Specifies the value that is paired with the specified name.

Required	Yes
Data type	String

Description:

Specifies a description of the name and value pair. The description should help to differentiate this pair for other defined pairs.

Required	No
Data type	Text area

Optional:

Specifies an optional attribute that determines whether this property must have a value.

Required	No
Data type	Boolean

Validation Expression:

Specifies a value that the administrative console and some host tools use to validate the contents of the value of this property.

Required	No
Data type	String

Data store [Settings]

The persistent store for messages and other state managed by the messaging engine.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Message store.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

UUID:

The universal unique identifier assigned by the system to this data store for administrative purposes.

Required	No
Data type	String

Data source JNDI name:

The JNDI name of the data source that the messaging engine uses to access the relational database management system (RDBMS) for the data store.

Required	Yes
Data type	Text

Schema name:

The name of the database schema used to contain the tables for the data store.

Each messaging engine stores its resources, such as tables, in a single schema. Each database schema is used by one messaging engine only. Although every messaging engine uses the same table names, its relationship with the schema gives each messaging engine exclusive use of its own tables.

Required	No
Data type	String

Authentication alias:

The name of the authentication alias used by the messaging engine to access the data source.

Required	No
Data type	drop-down list

Create tables:

If this option is selected, the messaging engine creates the database tables for the data store automatically. Otherwise, the database administrator must create the database tables manually.

Required	No
Data type	Boolean

Number of tables for permanent objects:

Number of database tables to use for storage of permanent objects.

Required	No
Data type	Integer
Range	1 through 2147483647

Number of tables for temporary objects:

Number of database tables to use for storage of temporary objects.

Required	No
Data type	Integer
Range	1 through 2147483647

Related Items

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

Default access roles [Settings]

This pane shows the role type assignments for the default access.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage default access roles.

security_value is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

This pane displays users and groups that have been assigned access roles for the default resource. Use this pane to add users and groups to the appropriate default role or roles, and to manage users and groups that have already been assigned role types for the default resource. Adding a user or group to any of the default roles grants the user or group the authorization permissions for that role for all the local destinations that are allowed to inherit defaults.

The information for each user and group is contained within a section that you can expand and collapse.

Expand/Collapse
Click to expand the section and display users and groups that have been assigned role types for the default resource.

General Properties

Select A check box that you can use to select the users and groups for which you want to manage access roles.

Name The name of the user or group that has an access role for the selected resource. If the user is a group member, the user ID and the group name is displayed.

Type The type of the user or group. There are three types of user or group: “user”, “group” and “member”. A user that inherits its access roles from a group has the type “member”.

Sender
Whether a user, group or member is in the sender role for a selected resource.

Receiver
Whether a user, group or member is in the receiver role for a selected resource.

Browser
Whether a user, group or member is in the browser role for a selected resource.

Creator
Whether a user, group or member is in the creator role for a selected resource.

Security access roles

In the administrative console, access role icons are used to represent whether a user or a group is in a particular access role. You can click an icon to add or remove selected users and groups to a particular access role for a selected resource.

An access role icon has three states:

- Access role type set.
- Access role type not set.
- Access role type inherited from group.

The following table describes how the access role icons represent these states, and how to change between them:

Table 214. Interacting with access role icons

Access role icon	Access role assignment state	User action
	Role type not set.	Click to change to role type set  .
	Role type set.	Click to change to role type not set. The icon changes to role type not set  if the user or group does not inherit access roles, or to role type inherited  if the role type does inherit access roles.
	Role type inherited from group.	Click to change to role type set  .
	Role type not set for a group. The group to which a user belongs does not have a role type.	Read only.
	Role type set for a group. The group to which a user belongs has a role type.	Read only.
	Role type not applicable.	Read only.

Buttons

Add	Click to add users and groups to this resource.
Remove	Click to remove selected users and groups from all the role types for this resource.

Destinations access roles [Collection]

This pane displays a list of all destinations known by the bus: aliases, foreign destinations, ports, queues, temporary destination prefixes, topic spaces and web services.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage destination access roles.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

This pane displays the name and type of all the bus destination defined for the selected service integration bus. Use this pane to view and change the access roles defined for selected destinations:

- To act on a single destination, click its name in the list.
- To act on more than one destination, first select the check box next to the name of each destination you want to act on, and then click **Manage Access Roles**.

Destination
The name of each destination defined for the selected bus.

Type The type of destination defined for the selected bus.

Buttons

Manage Access Roles	Click to view and manage users and groups assigned to the access role types for the selected resources.

Destinations access roles [Settings]

This pane displays the role type assignments for the selected destinations.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage destination access roles -> *destination_name* > Manage access roles.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

This pane lists users and groups that have been assigned role types for the destinations selected in the **Manage destination access roles** panel. You can use this pane to add and remove users and groups, and manage role type assignments and inheritance for a destination.

The information for each destination is contained within a section that you can expand and collapse. For each destination, a section header displays the type of destination, for example whether it is a queue, topic space, or some other type of destination and the destination name.

Expand/Collapse
Click the icon to expand the section and display the collection of users and groups that have been assigned role types for the selected destination.

Inherit from default
Select the check box to have the selected destination inherit access roles for users and groups from the default resource. Only access roles that apply to the selected destination can be inherited.

General Properties

Select A check box that you can use to select the users and groups for which you want to manage access roles.

Name The name of the user or group that has an access role for the selected resource. If the user is a group member, the user ID and the group name is displayed.

Type The type of the user or group. There are three types of user or group: “user”, “group” and “member”. A user that inherits its access roles from a group has the type “member”.

Sender
Whether a user, group or member is in the sender role for a selected resource.

Receiver
Whether a user, group or member is in the receiver role for a selected resource.

Browser
Whether a user, group or member is in the browser role for a selected resource.

Creator
Whether a user, group or member is in the creator role for a selected resource.

Security access roles

In the administrative console, access role icons are used to represent whether a user or a group is in a particular access role. You can click an icon to add or remove selected users and groups to a particular access role for a selected resource.

An access role icon has three states:

- Access role type set.
- Access role type not set.
- Access role type inherited from group.

The following table describes how the access role icons represent these states, and how to change between them:

Table 215. Interacting with access role icons

Access role icon	Access role assignment state	User action
	Role type not set.	Click to change to role type set  .
	Role type set.	Click to change to role type not set. The icon changes to role type not set  if the user or group does not inherit access roles, or to role type inherited  if the role type does inherit access roles.
	Role type inherited from group.	Click to change to role type set  .

Table 215. Interacting with access role icons (continued)

Access role icon	Access role assignment state	User action
	Role type not set for a group. The group to which a user belongs does not have a role type.	Read only.
	Role type set for a group. The group to which a user belongs has a role type.	Read only.
	Role type not applicable.	Read only.

Buttons

Add	Click to add users and groups to this resource.
Remove	Click to remove selected users and groups from all the role types for this resource.

Destinations [Collection]

A bus destination is defined on a service integration bus, and is hosted by one or more locations within the bus. Applications can attach to the destination as producers, consumers, or both to exchange messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The identifier by which this destination is known for administrative purposes.

Bus A service integration bus supports applications using message-based and service-oriented architectures. A bus is a group of interconnected servers and clusters that have been added as members of the bus. Applications connect to a bus at one of the messaging engines associated with its bus members.

Type Whether this bus destination is for a queue, topic space, or some other type of destination.

Description
An optional description for the bus destination, for administrative purposes.

Mediation
The name of the mediation that mediates this destination.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.
Mediate	Apply a mediation to a destination, to modify the destination handling of messages.

Unmediate	Remove a mediation from the destination, to remove the effect of the mediation from the destination handling of messages.

Destination defaults [Settings]

Properties to be applied when applications use destinations on this foreign bus when there is no explicit foreign destination definition.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Additional Properties] Destination defaults.

These properties apply when there is no explicit foreign destination or alias destination defined on the local bus to provide destination defaults.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Default priority:

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Integer
Range	0 through 9

Default reliability:

The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

This reliability setting is used if a message has not set a reliability or if the Enable producers to override default reliability is set to False.

- If an application is producing messages to a destination and that application does NOT specify a reliability for the message, then the Default reliability setting is used.
- If an application is producing messages to a destination and that application specifies a reliability, then the application reliability is ONLY applicable if the Enable producers to override default reliability property is set to True. Otherwise, the Default reliability setting is used.

Dynamic updates to this property are effective immediately.

Required	No
Data type	drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Maximum reliability:

The maximum reliability of messages accepted by this destination.

Producers cannot send messages to this destination with a reliability higher than the value specified for this property.

This attribute is not used for a temporary destination.

If the message exceeds the maximum reliability and the producer is local to the messaging engine, an exception is thrown.

If the message exceeds the maximum reliability and the producer is not local to the messaging engine, the message is sent to the exception destination that is defined for the target destination.

Dynamic updates to this property are effective immediately.

Required No
Data type drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to destinations on this foreign bus.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Boolean

Enable producers to override default reliability:

Select this option to enable producers to override the default reliability that is set on the destination.

- If an application is producing messages to a destination and that application does NOT specify a reliability for the message, then the Default reliability setting is used.
- If an application is producing messages to a destination and that application specifies a reliability, then the application reliability is ONLY applicable if the Enable producers to override default reliability property is set to True. Otherwise, the Default reliability setting is used.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Boolean

Include an RFH2 message header when sending messages to WebSphere MQ.:

If selected, messages sent to WebSphere MQ includes an RFH2 header. The RFH2 header stores additional information to that which is stored in theWebSphere MQ message header.

This property applies when a foreign bus connection represents a WebSphere MQ queue manager or queue-sharing group. When service integration converts a message from the service integration format to

WebSphere MQ format, by default it includes an MQRHF2 header in the WebSphere MQ message. This header contains message attributes, such as JMS message attributes, that are not WebSphere MQ message attributes and therefore do not appear in the WebSphere MQ message descriptor (MQMD). Some WebSphere MQ applications cannot process messages that include an MQRFH2 header. If messages sent to this destination will be processed by WebSphere MQ applications that cannot tolerate an MQRFH2, clear this option.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Boolean

Additional Properties

Context properties
Context information passed to the mediation.

File store [Settings]

The persistent store for messages and other state managed by the messaging engine.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Message store.

If you make any changes to the properties of a file store, you must restart the messaging engine to make those changes take effect.

For more information on the log file, permanent store file, and temporary store file and appropriate parameter values refer to ../ae/cjm1450_.dita.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

UUID:

The universal unique identifier assigned by the system to this data store for administrative purposes.

Required	No
Data type	String

Log size:

Size in megabytes of the log file.

Required	Yes
Data type	Long
Range	10 through 9223372036854775807

Log directory path:

Name of the log files directory.

When creating a messaging engine with a file store, you can select the Default log directory path radio button for this property. This option causes, the following default directory value to be used: `${USER_INSTALL_ROOT}/filestores/com.ibm.ws.sib/`*`messagingEngineName-messagingEngineUuid`*`/log` where *messagingEngineName* and *messagingEngineUuid* are substituted from the Name and UUID properties of the messaging engine.

After you have created the file store, the Log directory path field displays the directory path used for log files.

Important: When adding a cluster as a bus member, you must configure the log file to be on a file system that is accessible to all members of a cluster.

Required	No
Data type	Text

Minimum permanent store size:

The minimum number of megabytes reserved by each of the permanent store files. The permanent store files are never smaller than the log file.

For more information about the store files and appropriate values for the log file parameters refer to ../ae/cjm1450_.dita.

Required	Yes
Data type	Long
Range	0 through 9223372036854775807

Unlimited permanent store size:

Indicates whether the permanent store files are unlimited in size.

Required	No
Data type	Custom

Maximum permanent store size:

The maximum size in megabytes for the permanent store files.

Required	Yes
Data type	Long
Range	50 through 9223372036854775807

Permanent store directory path:

Name of the permanent store files directory.

When creating a messaging engine with a file store, you can specify a non-default directory to be used for permanent store files. To do this, select the **Permanent store directory path** radio button, then type a directory path value in the field provided.

When creating a messaging engine with a file store, you can select the **Default permanent store directory path** radio button for this property. This option causes the following default directory value to be used: `${USER_INSTALL_ROOT}/filestores/com.ibm.ws.sib/`*`messagingEngineName-messagingEngineUuid`*`/store` where *messagingEngineName* and *messagingEngineUuid* are substituted from the Name and UUID properties of the messaging engine.

After you have created the file store, the **Permanent store directory path** field displays the directory path used for permanent store files.

Important: When adding a cluster as a bus member, you must configure the permanent store file to be on a file system that is accessible to all members of a cluster.

Required	No
Data type	Text

Minimum temporary store size:

The minimum number of megabytes reserved by each of the temporary store files. The temporary store files are never smaller than the log file.

Required	Yes
Data type	Long
Range	0 through 9223372036854775807

Unlimited temporary store size:

Indicates whether the temporary store files are unlimited in size.

Required	No
Data type	Boolean

Maximum temporary store size:

The maximum size in megabytes for the temporary store files.

Required	Yes
Data type	Long
Range	50 through 9223372036854775807

Temporary store directory path:

Name of the temporary store files directory.

When creating a messaging engine with a file store, you can specify a non-default directory to be used for temporary store files. To do this, select the Temporary store directory path radio button, then type a directory path value in the field provided.

When creating a messaging engine with a file store, you can select the **Default temporary store directory path** radio button for this property. This option causes the following default directory value to be used: `${USER_INSTALL_ROOT}/filestores/com.ibm.ws.sib/`*`messagingEngineName`*`-`*`messagingEngineUuid`*`/store` where *messagingEngineName* and *messagingEngineUuid* are substituted from the Name and UUID properties of the messaging engine.

After you have created the file store, the **Temporary store directory path** field displays the directory path used for temporary store files.

Important: When adding a cluster as a bus member, you must configure the temporary store file to be on a file system that is accessible to all members of a cluster.

Required	No
Data type	Text

Foreign bus connections [Collection]

A foreign bus connection allows communication with another bus. The foreign bus can represent another Service Integration Bus, an instance of WebSphere MQ, or an indirect connection to another foreign bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

The physical link (also known as the routing type) between buses can be one of the following types:

- A *service integration bus link* from a messaging engine in the local bus to a messaging engine in the foreign bus.
- A *WebSphere MQ link* from a messaging engine in the local bus to a WebSphere MQ gateway queue manager. To the local bus the linked WebSphere MQ network appears as a foreign bus.
- An *indirect link*, which is a link that is made through one or more intermediate foreign buses.

Name The name of the bus with which this bus will exchange messages. This name must match exactly the name of the existing service integration bus that is defined as the foreign bus.

Routing type
Routing type defines the type of link used to make the connection to the foreign bus. Use a service integration bus link for a connection to a foreign service integration bus, a WebSphere MQ link for a connection to a foreign WebSphere MQ network, or an indirect routing if the connection to the foreign bus is via another foreign bus.

Description
An optional description for the foreign bus, for administrative purposes.

Configuration Status
Indicates that the configuration was deleted with remaining runtime data. Select this foreign bus connection (by clicking the **Name** column) to resolve the remaining runtime data.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.
Test connection	Test the foreign bus connection. This button does not work if the selected foreign bus uses a WebSphere MQ link that is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Foreign bus [Settings]

This pane shows the role type assignments for the selected foreign buses.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage foreign bus access roles -> *foreign_bus_name* > **Manage access roles**.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

This panel lists the users and groups that have been assigned access roles on the one or more selected foreign buses. Use this panel to add and remove users and groups to foreign bus access roles, and to manage existing role type assignments and inheritance for foreign buses.

The information for each foreign bus is contained within a section that you can expand and collapse. When collapsed, the section header displays the foreign bus name. When expanded, in addition to the header, the section displays an **Inherit from default** check box and a list of the users and groups that have been assigned role types on the foreign bus. Note that the only permitted role type for a foreign bus is Sender role.

Expand/Collapse
Click icon to expand the section and display a list of users and groups that have been assigned role types for the selected foreign bus.

Inherit from default
Select check box to have the selected foreign bus inherit access roles for users and groups from the default resource. Only access roles that apply to the selected foreign bus can be inherited.

General Properties

Select A check box that you can use to select the users and groups for which you want to manage access roles.

Name The name of the user or group that has an access role for the selected resource. If the user is a group member, the user ID and the group name is displayed.

Type The type of the user or group. There are three types of user or group: “user”, “group” and “member”. A user that inherits its access roles from a group has the type “member”.

Sender
Whether a user, group or member is in the sender role for a selected resource.

Security access roles

In the administrative console, access role icons are used to represent whether a user or a group is in a particular access role. You can click an icon to add or remove selected users and groups to a particular access role for a selected resource.

An access role icon has three states:

- Access role type set.
- Access role type not set.
- Access role type inherited from group.

The following table describes how the access role icons represent these states, and how to change between them:

Table 216. Interacting with access role icons

Access role icon	Access role assignment state	User action
	Role type not set.	Click to change to role type set .
	Role type set.	Click to change to role type not set. The icon changes to role type not set  if the user or group does not inherit access roles, or to role type inherited  if the role type does inherit access roles.
	Role type inherited from group.	Click to change to role type set .
	Role type not set for a group. The group to which a user belongs does not have a role type.	Read only.
	Role type set for a group. The group to which a user belongs has a role type.	Read only.
	Role type not applicable.	Read only.

Buttons

Add	Click to add users and groups to this resource.
Remove	Click to remove selected users and groups from all the role types for this resource.

Foreign bus connections [Settings]

A foreign bus connection allows communication with another bus. The foreign bus can represent another Service Integration Bus, an instance of WebSphere MQ, or an indirect connection to another foreign bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the bus with which this bus will exchange messages. This name must match exactly the name of the existing service integration bus that is defined as the foreign bus. Precise guidelines for naming a foreign bus connection can be found in the related concept topic Foreign buses.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this foreign bus for administrative purposes.

Required	No
Data type	String

Routing type:

Routing type defines the type of link used to make the connection to the foreign bus. Use a service integration bus link for a connection to a foreign service integration bus, a WebSphere MQ link for a connection to a foreign WebSphere MQ network, or an indirect routing if the connection to the foreign bus is via another foreign bus.

Required	No
Data type	String

Description:

An optional description for the foreign bus, for administrative purposes.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Text area

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to this foreign bus.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Boolean

Additional Properties

Destination defaults
Properties to apply when applications use destinations on this foreign bus and there is no explicit foreign destination or alias destination defined on the local bus to provide destination defaults.

Additional property for when the routing definition is not yet defined:

Create a routing definition
Select this link to define the routing definition for this foreign bus.

Additional property for direct service integration bus link foreign bus connections:

Service integration bus link routing properties
The routing properties for a service integration bus link to a foreign service integration bus.

Additional property for direct WebSphere MQ link foreign bus connections:

WebSphere MQ link routing properties
The routing properties for a link to a foreign bus that represents a WebSphere MQ network.

Additional property for indirect foreign bus connections:

Indirect routing properties
The routing definition for the next service integration bus in a sequence of connected buses.

Related Items

Related item for direct service integration bus link foreign bus connections:

Service integration bus links
The associated service integration bus links for this foreign bus.

Related item for direct WebSphere MQ link foreign bus connections:

WebSphere MQ links
The associated WebSphere MQ links for this foreign bus.

Foreign destination [Settings]

The name by which this foreign destination is known for administrative purposes.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *destination_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The identifier by which this destination is known for administrative purposes.

Type the name of the destination as configured on the foreign bus. This must match exactly the name of the target destination that exists in the foreign bus.

Required	No
Data type	String

Bus:

The name of the foreign bus on which this destination is defined.

This must match exactly the name of a foreign bus administrative object that is already defined on the bus on which this foreign destination is being created.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this destination for administrative purposes.

Required	No
Data type	String

Type:

Whether this bus destination is for a queue, topic space, or some other type of destination.

Required	No
Data type	String

Description:

An optional description for the bus destination, for administrative purposes.

Required	No
Data type	Text area

Enable producers to override default reliability:

Select this option to enable producers to override the default reliability that is set on the destination.

- If an application is producing messages to a destination and that application does NOT specify a reliability for the message, then the Default reliability setting is used.
- If an application is producing messages to a destination and that application specifies a reliability, then the application reliability is ONLY applicable if the Enable producers to override default reliability property is set to True. Otherwise, the Default reliability setting is used.

Required	No
Data type	Boolean

Default reliability:

The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

This reliability setting is used if a message has not set a reliability or if the SIBDestination.overrideOfQOSByProducerAllowed is set to False.

- If an application is producing messages to a destination and that application does NOT specify a reliability for the message, then the Default reliability setting is used.
- If an application is producing messages to a destination and that application specifies a reliability, then the application reliability is ONLY applicable if the Enable producers to override default reliability property is set to True. Otherwise, the Default reliability setting is used.

Required
No

Data type
drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Maximum reliability:

The maximum reliability of messages accepted by this destination.

Producers cannot send messages to this destination with a reliability higher than the value specified for this property.

Required
No

Data type
drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Default priority:

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

Required	No
Data type	Integer
Range	0 through 9

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

Required	No
Data type	Boolean

Include an RFH2 message header when sending messages to WebSphere MQ:

If selected, messages sent to WebSphere MQ include an RFH2 header. The RFH2 header stores additional information to that which is stored in theWebSphere MQ message header.

This property applies when the foreign bus that hosts the target destination is a WebSphere MQ queue manager or queue-sharing group. When service integration converts a message from the service integration format to WebSphere MQ format, by default it includes an MQRHF2 header in the WebSphere MQ message. This header contains message attributes, such as JMS message attributes, that are not WebSphere MQ message attributes and therefore do not appear in the WebSphere MQ message descriptor (MQMD). Some WebSphere MQ applications cannot process messages that include an MQRFH2 header. If messages sent to this destination will be processed by WebSphere MQ applications that cannot tolerate an MQRFH2, clear this option.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Boolean

Additional Properties

Context properties
Context information passed to the mediation.

Related Items

Application resources topology
A expandable tree view of all applications and messaging resources that reference the current destination.

Inbound messages [Collection]

The inbound message streams from the remote queue point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote queue points -> *identifier_name* -> [Additional Properties] Inbound messages**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This is the collection of inbound message streams by quality of service (priority and reliability) for messages being sent to this queue point from the remote messaging engine.

Priority
The stream priority.

Reliability
The stream reliability.

Number of messages
The current number of messages on the stream.

Last delivered message sequence ID
The sequence ID of the last message delivered.

Status
The status of the stream.

Messages
View the messages on the stream.

Indirect routing properties [Settings]

The routing definition for the next service integration bus in a sequence of connected buses.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Additional Properties] Indirect routing properties.

You configure an indirect link when a service integration bus needs to communicate with a foreign bus through one or more intermediate buses.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Next bus in route:

The name of the next service integration bus in the sequence of connected buses.

An intermediate bus can be a WebSphere MQ system rather than a service integration bus.

Required	No
Data type	drop-down list

JFAP inbound channel [Settings]

A channel which can be used in combination with the TCP Channel - or other channels that support the same application interface to accept inbound connections to a WebSphere system integration bus messaging engine.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports -> *chain_name* > **JFAP inbound channel**.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Transport channel name:

Specifies the unique name for a given layer in a network protocol stack.

Required	Yes
Data type	String

Discrimination weight:

Specifies the discrimination weight that is used to determine the order in which the channels obtain access to the incoming connection if the transport channels are shared amongst several transport chains. The transport channel with the lowest discrimination weight has the first opportunity to accept the incoming connection.

Required	No
Data type	Integer
Range	0 through 2147483647

Heart beat interval:

The amount of idle time, in seconds, before the channel attempts to solicit a response from its peer to check that the peer is still active. The sum of this value and the heartbeat timeout value determines the maximum amount of time that it can take to discover that some types of network failure have occurred.

Required	No
Data type	Integer
Range	0 through 2147483647

Heart beat timeout:

The amount of time, in seconds, to wait for a response from a peer after deciding to check that the peer is still alive.

Required	No
Data type	Integer
Range	0 through 2147483647

Additional Properties

Custom properties

Specifies additional custom properties for this runtime component. Some components use custom configuration properties that can be defined here.

Known link transmitter inbound streams [Collection]

This pane displays the inbound message streams from messaging applications that are connected to the remote messaging engine, and that are producing messages to this WebSphere MQ link sender channel transmitter.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel transmitters -> *WebSphere_MQ_sender_channel_name* -> Known link transmitters -> *messaging_engine_name***
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> Known link transmitters -> *messaging_engine_name***

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Priority
The priority of the messages on the given inbound stream.

Reliability
The reliability of the messages on the given inbound stream.

Number of messages
The number of messages pending receipt on the given inbound stream.

Last delivered message sequence
The sequence identifier of the last message delivered from the inbound stream to the sender channel transmitter.

Status
The runtime status of the inbound stream. Green (running), Amber (running but a problem exists), Red (stopped).

Messages
Select this link to view the message collection for messages on the given inbound stream.

Known link transmitter stream messages [Collection]

This pane displays current messages on the inbound message stream.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel transmitters -> *WebSphere_MQ_sender_channel_name* -> Known link transmitters -> *messaging_engine_name* -> Messages**
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> Known link transmitters -> *messaging_engine_name* -> Messages**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Maximum messages displayed
The maximum number of messages retrieved and displayed.

Position
The position on the inbound stream from the known link transmitter.

Identifier
The identifier of the message. Select this link to view the message detail panel for the selected message.

Previous sequence identifier
The identifier of the previous message on the queue point.

State The state of the message on the inbound stream (“Awaiting delivery”).

Buttons

Refresh	Refresh the collection with the current set of messages on the inbound stream. Only the number of messages up to the specified **Maximum messages displayed** are retrieved and displayed.

Known link transmitters [Collection]

This pane displays the inbound receiver queues for messaging applications connected to remote messaging engines, that are producing messages to this WebSphere MQ link sender channel transmitter.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel transmitters -> *WebSphere_MQ_sender_channel_name* -> Known link transmitters**
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> Known link transmitters**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Messaging engine
The messaging engine that has a link transmitter that is transmitting messages to this sender channel transmitter. Select the link to view the inbound streams for the messages from the known link transmitter.

Current inbound messages
The total number of messages currently pending receipt on the inbound streams from this known link transmitter.

Inbound messages received
The total number of messages that have been received on the inbound streams from this known link transmitter since the messaging engine started.

Known remote publication points [Collection]

The remote messaging engines that have remote producers connected to this publication point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message Points] Publication points** > ***identifier_name* -> [Additional Properties] Known remote publication points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This collection represents all the remote publication points at remote messaging engines where there are applications or clients connected that are publishing messages to this local publication point.

Messaging engine
The messaging engine where the remote publication point is located.

Current inbound messages
The current number of queued inbound messages for the publication point.

Inbound messages received
The total number of messages received on this publication point since the messaging engine started.

Known remote publication points [Settings]

The remote messaging engines that have remote producers connected to this publication point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message Points] Publication points** > ***identifier_name*** **-> [Additional Properties] Known remote publication points -> *messaging_engine_name***.

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Messaging engine:

The messaging engine where the remote publication point is located.

Required	No
Data type	String

Current inbound messages:

The current number of queued inbound messages for the publication point.

Required	No
Data type	String

Inbound messages received:

The total number of messages received on this publication point since the messaging engine started.

Required	No
Data type	String

Additional Properties

Inbound messages
The inbound message streams from the remote publication point.

Queue [Settings]

A queue for point-to-point messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The identifier by which this destination is known for administrative purposes.

Do not use an underscore character "_" as the first character, because this naming convention is reserved for system use.

You can use a naming convention to suggest a hierarchical structure for destinations; for example, by using dotted notation for a destination name `Library.Shelf.Book1A`. Such structure can be useful for organizing queues into logical groups for ease of association, and to permit the use of wildcard notation in filters; for example, `Library.Shelf.*` or `*.Queue.accountXYZ`.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this destination for administrative purposes.

Required	No
Data type	String

Type:

Whether this bus destination is for a queue, topic space, or some other type of destination.

A queue for point-to-point messaging.

Required	No
Data type	String

Description:

An optional description for the bus destination, for administrative purposes.

Required No
Data type Text area

Mediation:

The name of the mediation that mediates this destination.

Required No
Data type String

Enable producers to override default reliability:

Select this option to enable producers to override the default reliability that is set on the destination.

Required No
Data type Boolean

Default reliability:

The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

Required No
Data type drop-down list
Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Maximum reliability:

The maximum reliability of messages accepted by this destination.

Required No

Data type drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Default priority:

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

Required No
Data type Integer
Range 0 through 9

Exception destination:

Use these properties to define what happens to any messages that cannot be delivered to this destination.

None:

The bus destination does not use an exception destination and undeliverable messages are not rerouted to an exception destination.

Attempts to redeliver the message continue, up to the maximum failed deliveries limit set for the bus destination. Then, attempts to redeliver the message continue with a time interval between retry attempts. This interval is either the Default blocked destination retry interval of the messaging engine that is associated with this destination, or the Blocked retry timeout that is set for this destination. The Default blocked destination retry interval value can be used by all queue and topic destinations associated with this messaging engine. To set a time interval specifically for this destination, select **Override messaging engine blocked retry timeout default**, then enter a blocked retry timeout value for this destination.

Required No
Data type Radio button
Default Not selected

Override messaging engine blocked retry timeout default:

Override the blocked queue retry interval configured on the messaging engine owning the destination.

Select this property to set the blocked retry timeout for this destination. This property is available only when **None** is selected for the exception destination.

Required	Yes, if **None** is selected.
Data type	Boolean
Default	Unchecked. The Default blocked destination retry interval value of the associated messaging engine is used.

Blocked retry timeout in milliseconds:

When no exception destination is configured, the time interval to apply between retry attempts, after the maximum failed deliveries limit is reached, for this destination.

This property is available only when **Override messaging engine blocked retry timeout default** is selected in the exception destination properties.

Required	Yes, if Override messaging engine blocked retry timeout default is checked.
Data type	Integer

System:

The bus destination uses the system default exception destination.

Undeliverable messages are routed to the system default exception destination of the messaging engine that detects the problem: _SYSTEM.Exception.Destination.*messaging_engine_name*.

Required	No
Data type	Radio button
Default	Selected

Specify:

The bus destination uses the exception destination specified in this field.

The exception destination must be a queue, on the same bus or a foreign bus, and must exist when the exception destination processing is configured.

Required	No
Data type	Radio button
Default	Not selected

Maximum failed deliveries per message:

The maximum number of failed attempts to process a message. After this number of failed attempts, if an exception destination is configured, the message is forwarded from the intended destination to its exception destination. If an exception destination is not configured, a time interval between retry attempts is applied.

This interval is either the Default blocked destination retry interval of the messaging engine that is associated with this destination, or the Blocked retry timeout that is set for this destination.

Required	Yes, if an exception destination has been configured.
Data type	Integer
Default	5
Range	0 through 2147483647

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

Required	No
Data type	Boolean

Receive allowed:

Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination.

Required	No
Data type	Boolean

Receive exclusive:

Select this option to allow only one consumer to attach to each message point. If this option is not selected multiple consumers will be allowed to attach and receive messages from each message point.

Required	No
Data type	Boolean

Maintain strict message order:

Enabling this option will maintain the strict ordering of messages for this destination.

Required	No
Data type	Boolean

Reply destination:

The name of a destination to be appended to any non-empty reverse routing path of messages sent to this destination.

This property is intended for use with mediations on reply messages.

For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

Required	No
Data type	String

Reply destination bus:

The bus on which the reply destination exists.

This property is intended for use with mediations on reply messages.

For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

Required	No
Data type	String

Default forward routing path:

The value to which a message's forward routing path will be set if the message contains no forward routing path. This identifies a sequential list of intermediary bus destinations that messages must pass through to reach a target bus destination. The format of the field is a list of line-delimited bus destinations specified as *bus*:*name*.

If you want to forward messages to one or more bus destinations, type a list of bus destinations. Type each destination entry on a separate line, and in the form *bus_name*:*destination_name* or :*destination_name*

Where

bus_name
: Is the name of the service integration bus on which the destination is configured. If you do not specify a bus name, the destination is assumed to be on the same bus as the destination for which you are setting this property.

destination_name
: is the name of a bus destination.

Required	No
Data type	Text area

Message points

Queue points
: Displays a list of queue points used to hold messages pending delivery to receiving applications.

Mediation points
: Displays a list of mediation points created when a mediation is associated with a bus destination.

Additional Properties

Context properties
: Context information passed to the mediation.

Mediation execution points
: Mediation execution points for the processing of messages from mediation message points that are on a WebSphere MQ server.

Related Items

Application resources for this destination
: This pane provides an expandable tree view of all the applications and messaging resources that reference the current destination, both directly and indirectly. As many of the references as possible are resolved to links to the associated configuration panel for the referenced object.

  Use this panel to inspect the configuration from the queue or topic space (destination) to the application or other JMS resources to ensure that the configuration is correct.

Remote queue points [Collection]

The remote queue points that are producing or consuming messages to or from queue points on remote messaging engines.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote queue points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This panel applies only to WebSphere Application Server Network Deployment or WebSphere Application Server for z/OS, with multiple messaging engines.

A remote queue point represents a remote proxy for a queue point that exists on a remote messaging engine (any other messaging engine on this bus). Messages are sent to or received from an application or client connected to the remote messaging engine. The remote queue point manages an outbound message stream to the queue point on the remote messaging engine, and an inbound message retrieval request stream for messages received from the queue point.

This collection gives an indication of the applications using these remote queue points, where the applications are connected and the associated message flow from each messaging engine.

Identifier
The remote queue point identifier.

Messaging engine
The remote messaging engine where this queue point is localized.

Current outbound messages
The current number of outbound messages queued to the queue point.

Outbound messages sent
The total number of messages sent to the queue point since the messaging engine started.

Current message requests
The current number of active message retrieval requests sent to the queue point from this remote queue point.

Completed message requests
The total number of completed message retrieval requests sent to the queue point from this remote queue point since the messaging engine started.

Buttons

Delete all messages	Delete all messages from the outbound message stream of the selected remote queue point. Messages in “Pending Acknowledgment” state might already have been received at the remote queue point. You cannot delete messages in “Committing” state; you must first commit their transactions.

Move all messages	Move all messages from the outbound message stream of the selected remote queue point to the exception destination configured for the destination. Messages in "Pending Acknowledgment" state might already have been received at the remote queue point. Moving these messages to the exception destination results in two copies of the message in the bus. You cannot move messages in "Committing" state; you must first commit their transactions.

Known remote queue points [Collection]

The remote messaging engines that have remote producers or consumers connected to this queue point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message Points] Queue points** > ***identifier_name*** **-> Known remote queue points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This collection represents all the remote queue points at remote messaging engines that have applications or clients connected, where those applications or clients are either sending messages to or receiving messages from this local queue point.

Messaging engine
The messaging engine where the remote queue point is located.

Current inbound messages
The current number of queued inbound messages for the queue point.

Inbound messages received
The total number of messages received on this queue point since the messaging engine started.

Current messages requests
The current number of active message retrieval requests from the remote queue point.

Completed messages requests
The total number of completed message retrieval requests from the remote queue point since the messaging engine started.

Remote queue points [Settings]

The remote queue points that are producing/consuming messages to/from queue points on remote messaging engines.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote queue points -> *identifier_name*** .

- "Runtime tab" on page 2082

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The remote queue point identifier.

Required	No
Data type	String

Messaging engine:

The remote messaging engine where this queue point is localized.

Required	No
Data type	String

Current outbound messages:

The current number of outbound messages queued to the queue point.

Required	No
Data type	String

Outbound messages sent:

The total number of messages sent to the queue point since the messaging engine started.

Required	No
Data type	String

Current message requests:

The current number of active message retrieval requests sent to the queue point from this remote queue point.

Required	No
Data type	String

Completed message requests:

The total number of completed message retrieval requests sent to the queue point from this remote queue point since the messaging engine started.

Required	No
Data type	String

Message requests issued:

The total number of active and completed message retrieval requests sent to the queue point from this remote queue point since the messaging engine started.

Required	No
Data type	String

Additional Properties

Outbound messages
The outbound message streams from this remote message point to the remote localized message point.

Message Requests
A snapshot of the current message retrieval requests from the message point.

Known remote queue points [Settings]

The remote messaging engines that have remote producers or consumers connected to this queue point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message Points] Queue points** > ***identifier_name*** **-> Known remote queue points -> *messaging_engine_name*** .

- “Runtime tab”

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Messaging engine:

The messaging engine where the remote queue point is located.

Required	No
Data type	String

Current inbound messages:

The current number of queued inbound messages for the queue point.

Required	No
Data type	String

Inbound messages received:

The total number of messages received on this queue point since the messaging engine started.

Required	No
Data type	String

Current messages requests:

The current number of active message retrieval requests from the remote queue point.

Required	No
Data type	String

Completed messages requests:

The total number of completed message retrieval requests from the remote queue point since the messaging engine started.

Required	No
Data type	String

Message requests received:

The total number of active and completed message retrieval from the remote queue point since the messaging engine started.

Required	No
Data type	String

Additional Properties

Inbound messages
The inbound message streams from the remote queue point.

Message requests
The message retrieval requests from the remote queue point.

Known remote subscription points [Collection]

The remote messaging engines that have remote consumers connected to this subscription point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message points] Publication points** > ***identifier_name*** **-> [Additional Properties] Known remote subscriptions**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This collection represents all the remote subscription points (at remote messaging engines) where there are applications or clients connected that are receiving messages from a subscription that has its home on this local topic space.

Messaging engine
The messaging engine where the remote subscription point is located.

Current messages requests
The current number of active message retrieval requests from the remote subscription point.

Completed messages requests
The total number of completed message retrieval requests from the remote subscription point since the messaging engine started.

Known remote subscription points [Settings]

The remote messaging engines that have remote consumers connected to this subscription point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message points] Publication points** > ***identifier_name*** **-> [Additional Properties] Known remote subscriptions -> *messaging_engine*** .

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Messaging engine:

The messaging engine where the remote subscription point is located.

Required	No
Data type	String

Current messages requests:

The current number of active message retrieval requests from the remote subscription point.

Required	No
Data type	String

Completed messages requests:

The total number of completed message retrieval requests from the remote subscription point since the messaging engine started.

Required	No
Data type	String

Message requests received:

The total number of active and completed message retrieval from the remote subscription point since the messaging engine started.

Required	No
Data type	String

Additional Properties

Message requests
The message retrieval requests from the remote subscription point.

Link receiver stream messages [Collection]

This pane displays the current messages on the link receiver stream.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links -> *link_name* -> [Related Items] Link receivers -> *foreign_messaging_engine* -> Messages

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Position
The position in the queue of any pending messages that are waiting to be received.

Identifier
The identifier of this message.

Previous sequence identifier
The sequence identifier of the previous message.

State The state of the given message ("Awaiting delivery").

Target bus
The target bus for the message.

Target destination
The target destination for the message.

Time on queue
The time since the message arrived on this messaging engine.

Link receiver streams [Collection]

This pane displays the inbound message streams from messaging applications on the foreign bus for the link receiver.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links -> *link_name* -> [Related Items] Link receivers -> *foreign_messaging_engine***
-

- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Service integration bus link receivers**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Priority
The priority of the messages on the given stream.

Reliability
The reliability of the messages on the given stream.

Current inbound messages
The number of messages pending receipt on this message stream.

Messages received
The number of messages received on this message stream since the messaging engine started.

Last delivered message sequence ID
The sequence identifier of the last message to be received.

Status
The status of the link receiver stream. Green (running), Amber (running but a problem exists), Red (stopped).

Messages
A link to view the messages on this link receiver stream.

Link receivers [Collection]

A link can have multiple link receivers. For applications that use point-to-point messaging, there is one link receiver for each messaging engine in the foreign bus, that is, the bus that sends messages across the link. For applications that use publish/subscribe messaging, there is one link receiver for each topic space in the foreign bus. The link receiver acts as an inbound receiver queue for a messaging engine that has applications attached and that is producing messages across this service integration bus link.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Service integration bus links -> *link_name* -> [Related Items] Link receivers**
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links -> *link_name* -> [Related Items] Link receivers**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Foreign messaging engine
The name of the messaging engine in the foreign bus that sends messages across the service integration bus link.

Type The message domain type of this link receiver, that is, point-to-point or publish/subscribe.

Topic space
For a publish/subscribe link receiver, the topic space from which this link receiver is receiving messages across the link. This field does not apply to point-to-point link receivers.

Status
The status of the link receiver. Green (running), Amber (running but a problem exists), Red (stopped).

Current inbound messages
The current number of messages pending receipt on the link receiver.

Messages received
The total number of messages received on the link receiver since the messaging engine started.

Time since last message received
The time since the last message was received on the link receiver or since the messaging engine started.

Link transmitters [Collection]

A link can have multiple link transmitters. For applications that use point-to-point messaging, there is one link transmitter on each messaging engine in the source bus. For applications that use publish/subscribe messaging, there is one link transmitter for each topic space in the source bus. The link transmitter acts as a transmission queue where produced messages are persisted before transmission across the service integration bus link or WebSphere MQ link to the foreign bus.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links -> *link_name* -> [Related Items] Link transmitters**
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Related Items] Link transmitters**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Related Items] Link transmitters**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> [Related Items] Link transmitters**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Messaging engine
The name of the messaging engine that hosts the link transmitter.

Type The message domain type of this link transmitter, that is, point-to-point or publish/subscribe.

Topic space
For a publish/subscribe link transmitter, the topic space to which this link transmitter is transmitting messages across the link. This field does not apply to point-to-point link transmitters.

Status
The status of the link transmitter. Green (running), Amber (running but a problem exists), Red (stopped).

Current outbound messages
The number of currently queued messages that are to be transmitted.

Messages sent
The number of messages sent from this link transmitter since the messaging engine started.

Time since last message sent
The time since the last message was transmitted over the link or the messaging engine was started.

Buttons

Delete all messages	Delete all available messages that are queued for the selected link transmitters.
Move all messages	Move all available messages that are queued for the selected link transmitters to the messaging engine exception destination that is local to the link transmitter.

Link transmitter stream messages [Collection]

This pane displays the outbound messages to the foreign bus link, on the link transmitter stream.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links -> *link_name* -> [Related Items] Link transmitters -> *messaging_engine* -> Messages**
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Related Items] Link transmitters -> *messaging_engine_name* -> Messages**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Related Items] Link transmitters -> *messaging_engine_name* -> Messages**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> [Related Items] Link transmitters -> *messaging_engine_name* -> Messages**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ link transmitters -> *link_name* -> Messages**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Position
The position of the message in the queue on this message stream.

Identifier
The message identifier.

State The state of the queued message.

Committing
The message is currently being added under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction. While this message is in this state, no subsequent messages will be accepted by the queue point for which it is intended. There might be a build up of messages on this remote queue point.

Complete
Transmission of the message to the queue point has been completed. The local copy of the message is deleted automatically.

Pending send
The message is waiting to be sent to the queue point. Messages will be in this state if either of the following conditions apply:

- The messaging engine that owns the queue point is not currently available or accepting messages (for example, if its queue point has reached its high message threshold).
- The queue point already has a significant number of messages pending acknowledgement.

The message is sent when the condition no longer applies.

Pending acknowledgement
The message has been transmitted to the queue point and the messaging engine is waiting for an acknowledgement that the queue point has received the message correctly. If this state persists, the messaging engine to which the message was transmitted might not be reachable or a previous message on the remote queue point is still committing, and needs resolving.

Transaction ID
The transaction identifier if the message is locked within a transaction.

Target bus
The name of the target bus.

Target destination
The name of the target bus destination.

Approximate message length (bytes)
The approximate length of the message in bytes.

Buttons

Move	Move the selected messages to the local exception destination of the link transmitter.
Delete	Delete the selected messages.

Link transmitter streams [Collection]

This pane displays the outbound message streams, by quality of service, for the link transmitter.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links -> *link_name* -> [Related Items] Link transmitters -> *messaging_engine***
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Related Items] Link transmitters -> *messaging_engine_name***
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Related Items] Link transmitters -> *messaging_engine_name***
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> [Related Items] Link transmitters -> *messaging_engine_name***

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Priority
The priority of the messages on the given stream.

Reliability
The reliability of the messages on the given stream.

Number of messages queued
The number of messages on this message stream.

Number of messages sent
The number of messages sent from this message stream since the messaging engine started.

Status
The status of the message stream. Green (running), Amber (running but a problem exists), Red (stopped).

Messages
A link to view the messages on this message stream.

Manage foreign bus access roles [Collection]

A foreign bus is another bus with which this bus can exchange messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage foreign bus access roles.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

This pane displays the names of all the foreign buses defined on the selected bus. Use this pane to view and change the access roles defined for selected buses:

- To view and change the access roles defined for a single foreign bus, click the foreign bus name.
- To view and change the access roles defined for more than one foreign bus, select the check box in the **Select** column for each foreign bus, and click **Manage Access Roles**.

Foreign bus
The name of each foreign bus defined for the selected bus.

Buttons

Manage Access Roles	Click to view and manage users and groups assigned to the access role types for the selected resources.

Mediation points [Collection]

Bus member (server or cluster) where the mediations for the destination run.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Status
The runtime status of the mediation point.

Reason

Buttons

Start	Start selected items.
Stop	Stop selected items.

Mediation points [Collection]

Bus member (server or cluster) where the mediations for the destination run.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points -> Runtime.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Status
The runtime status of the mediation point.

Reason

Buttons

Start	Start selected items.
Stop	Stop selected items.

Mediation points [Settings]

Bus member (server or cluster) where the mediations for the destination run.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points -> *mediation_point_name*.

- "Configuration tab"
- "Runtime tab" on page 2094

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Buttons

Refresh	Refresh the number of messages.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this message point for administrative purposes.

Required No
Data type String

High message threshold:

A threshold above which the messaging system will take action to limit the addition of more messages to this message point.

Required No
Data type Long
Range 1 through 9223372036854775807

Send allowed:

Clear this option (setting it to false) to stop messages from being put onto this message point. This value will be overridden by the parent destination if that destination has sendAllowed disabled, which stops messages from being put onto all its message points.

Required No
Data type Boolean

Initial state:

Whether the mediation point is started or stopped when the hosting messaging engine is first started. Until started, the mediation point is unavailable.

Required No
Data type drop-down list
Range

Started
When the associated messaging engine is started, the mediation is started and is available to process messages.

Stopped
When the associated messaging engine is started, the mediation is stopped and is not available to process messages.

Target UUID:

The UUID of the bus destination for which this is a message point.

Required No
Data type String

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime

property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Buttons

Refresh	Refresh the number of messages.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required No
Data type String

Run-time ID:

The internal runtime identifier assigned to this message point.

Required No
Data type String

High message threshold:

A threshold above which the messaging system will take action to limit the addition of more messages to this message point.

Required No
Data type String

Send allowed:

Clear this option (setting it to false) to stop messages from being put onto this message point. This value will be overridden by the parent destination if that destination has sendAllowed disabled, which stops messages from being put onto all its message points.

Required No
Data type Boolean

Status:

The runtime status of the mediation point.

Required No
Data type drop-down list

Range

Waiting
The mediation is waiting to start. This might be because the application server is not yet open for e-business, or because a previous instance of the mediation has not yet been deleted.

Started
The mediation is started and is available to process messages.

Stopping
The mediation is in the process of stopping.

Stopped
The mediation is stopped. The reason why the mediation is stopped is shown in the Reason attribute.

Deleting
The mediation is in the process of being deleted.

Reason:

Required	No
Data type	Text area

Current message depth:

The number of messages on the message point.

Required	No
Data type	String

Additional Properties

Messages
Messages queued on the mediation point.

Mediation thread pool [Settings]

The thread pool used to allocate threads for the execution of mediation handlers.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Mediation thread pool.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Minimum size:

Minimum size of thread pool.

Required	No
Data type	Integer
Range	0 through 2147483647

Maximum size:

Maximum size of thread pool.

Required	No
Data type	Integer
Range	0 through 2147483647

Thread inactivity timeout:

Specifies the number of milliseconds of inactivity that should elapse before a thread is reclaimed.

Required	No
Data type	Integer
Range	0 through 2147483647

Allow thread allocation beyond maximum thread pool size:

Specifies whether the number of threads can increase beyond the maximum number configured for the thread pool.

Required	No
Data type	Boolean

Mediations [Collection]

A mediation that is associated with a bus destination to apply processing to messages on that destination.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Mediation name
The name by which this mediation is known for administrative purposes.

Handler list name
The name of the handler list that was defined when the mediation was deployed.

Description
An optional description for the mediation, for administrative purposes.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Mediations [Settings]

A mediation that is associated with a bus destination to apply processing to messages on that destination.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Mediations -> *mediation_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Mediation name:

The name by which this mediation is known for administrative purposes.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this mediation for administrative purposes.

Required	No
Data type	String

Description:

An optional description for the mediation, for administrative purposes.

Required	No
Data type	Text area

Handler list name:

The name of the handler list that was defined when the mediation was deployed.

Required Yes
Data type String

Global transaction:

Whether or not a global transaction is started for each message processed.

Cleared
A local transaction is started for each message processed. You only need to select this option for mediations that access other resource managers such as databases, or interact with Enterprise JavaBeans that require a global transaction.

Selected
A global transaction is started for each message processed.

Required No
Data type Boolean

Allow concurrent mediation:

Select this option (setting it to true) to apply the mediation to multiple messages concurrently. Message ordering is not preserved. The default option is false.

Selected
Apply the mediation to multiple messages concurrently, and preserve message ordering.

Cleared
Apply the mediation to a single message at a time. This setting is required to ensure that message ordering is preserved.

Required No
Data type Boolean

Selector:

Controls which messages are sent to the mediation. If a message matches the rule defined by the selector text string, then the mediation is applied to the message.

If the message does not match the rule defined by the selector text string, then the message is not mediated. If a message contains both Selector and Discriminator, it must match both rules for the message to be mediated. If either the Selector or the Discriminator rule does not match, the message is not mediated.

You should base the content of the selector text string on an understanding of which messages should be processed by the mediation. The format of the selector string is the same as for JMS selectors.

Required No
Data type String

Discriminator:

Controls which messages have the mediation applied to them. If the topic of a message matches the rule specified by the discriminator text string, then the mediation is applied to the message. If both the selector and discriminator are specified, the message must match both rules for the mediation to be applied to the message.

Compare this property with the Selector property. The rule specified by the Selector examines the header and properties of the message, whereas the discriminator examines the topic of the message. If a message contains both Selector and Discriminator, it must match both rules for the message to be mediated. If either the Selector or the Discriminator rule does not match, the message is not mediated.

You should base the content of the discriminator text string on an understanding of which message topics should be processed by the mediation. The format of the discriminator is the same as the topic discriminator specification.

Required	No
Data type	String

Additional Properties

Context properties
Context information passed to the mediation.

Message body [Settings]

The contents of the message body.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points -> *queue_point_name* -> Runtime > **Messages -> *message_name* -> Message body**.

Message points [Collection]

Queue points and publication points for the messaging engine.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] Message points.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Destination type
Whether the message point is a queue or topic space.

Message Requests [Collection]

A snapshot of the current message retrieval requests from the message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > [Remote Message Points] Remote publication points > *identifier_name* -> [Inbound Properties] Remote subscriptions -> *subscription_name* -> [Additional Properties] Message requests.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

The collection represents the currently active message retrieval requests by applications or clients connected to this messaging engine, for messages on the message point at the remote messaging engine. A message retrieval request represents a request from a client or application connected to one messaging engine, for a message from a queue point or a subscription on a different messaging engine. When the remote messaging engine has satisfied the message retrieval request, the application or client receives the message for processing.

Request ID
The request identifier.

Times out at
The time at which the message retrieval request will timeout.

Selector
The retrieval request message selection criteria.

Status
The state of the message retrieval request.

Message requests [Collection]

A snapshot of the current message retrieval requests from the remote message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > [Remote message points] Remote queue points -> *identifier_name* -> [Additional Properties] Message requests.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

The collection represents the currently active message retrieval requests for messages on this message point, by applications or clients connected to the remote messaging engine. When a message satisfying a message retrieval request becomes available on this message point, the message is exclusively assigned to the request and sent to the remote messaging engine to be delivered to the requesting application. If the requesting application accepts the message, the local message is deleted from the message point. If the message is rejected, the local message is made available again for other applications to request.

Request ID
The request identifier.

Times out at
The time at which the message retrieval request will timeout.

Selector
The retrieval request message selection criteria.

State The state of the message retrieval request.

Buttons

Cancel request	Cancel the selected message retrieval request and make any message that is allocated to the request available to other application requests.
Cancel request and delete message	Cancel the selected message retrieval request and delete any message that is allocated to the request.

Messages [Collection]

A snapshot of the current outbound messages for the message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote queue points -> *identifier_name* -> [Additional Properties] Outbound messages**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Sequence ID
The message sequence identifier.

API message ID
The API message identifier.

Time produced
The time the message was produced.

State The state of the queued message.

Committing
The message is currently being added under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction. While this message is in this state, no subsequent messages will be accepted by the queue point for which it is intended. There might be a build up of messages on this remote queue point.

Complete
Transmission of the message to the queue point has been completed. The local copy of the message is deleted automatically.

Pending send
The message is waiting to be sent to the queue point. Messages will be in this state if either of the following conditions apply:

- The messaging engine that owns the queue point is not currently available or accepting messages (for example, if its queue point has reached its high message threshold).
- The queue point already has a significant number of messages pending acknowledgement.

The message is sent when the condition no longer applies.

Pending acknowledgement
The message has been transmitted to the queue point and the messaging engine is waiting for an acknowledgement that the queue point has received the message correctly. If this state persists, the messaging engine to which the message was transmitted might not be reachable or a previous message on the remote queue point is still committing, and needs resolving.

Buttons

Delete	Delete the selected items.
Move	Move the selected outbound messages to the exception destination configured for the destination. Messages in "Pending Acknowledgment" state might already have been accepted to the remote queue point. Moving these messages to the exception destination results in two copies of the message in the bus. You cannot move messages in "Committing" state; you must first commit their transaction.

Messages [Collection]

A snapshot of the current outbound messages for the message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote publication points -> *identifier_name* -> [Outbound Properties] Messages**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Sequence ID
The message sequence identifier.

API message ID
The API message identifier.

Time produced
The time the message was produced.

State The state of the queued message.

Committing
The message is currently being added under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction. While this message is in this state, no subsequent messages will be accepted by the queue point for which it is intended. There might be a build up of messages on this remote queue point.

Complete
Transmission of the message to the queue point has been completed. The local copy of the message is deleted automatically.

Pending send
The message is waiting to be sent to the queue point. Messages will be in this state if either of the following conditions apply:

- The messaging engine that owns the queue point is not currently available or accepting messages (for example, if its queue point has reached its high message threshold).
- The queue point already has a significant number of messages pending acknowledgement.

The message is sent when the condition no longer applies.

Pending acknowledgement
The message has been transmitted to the queue point and the messaging engine is waiting for an acknowledgement that the queue point has received the message correctly. If this state persists, the messaging engine to which the message was transmitted might not be reachable or a previous message on the remote queue point is still committing, and needs resolving.

Buttons

Delete	Delete the selected items.

Messages [Collection]

The messages on the message point.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points -> *queue_point_name* -> Runtime** > **Messages**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Message points] Mediation points -> *mediation_point_name* -> Runtime** > **Messages**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A snapshot of the set of messages queued on the message point. You can display or remove selected messages from the list.

Because this is a snapshot of the message point, a message listed might no longer exist when you attempt to display it or act on it.

Position
The position of the message on the message point list.

Messaging engine message identifier
Messaging engine message identifier.

State The current state of the message related to a transaction that the message is part of. If the message is part of a transaction, the transaction identifier is shown in the Transaction ID field. The possible states are:

Available
The message is available for consumption.

Locked
The message is currently unavailable. The message is in this state temporarily, possibly because it is being consumed by a non-transacted consumer.

Remote lock
The message is currently locked to a consumer attached to another, remote, messaging engine in the bus. The message will remain locked until the remote messaging engine responds with a decision on the message. If the remote messaging engine is stopped, the message will remain locked until the messaging engine is restarted. A corresponding message request for a “known remote queue point” will identify the remote messaging engine that is making the request.

Removing
The message is currently being removed under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Committing
The message is currently being added under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Pending retry
The message is currently unavailable before being eligible for a retry. This might be because a message-driven bean is configured to delay failing message retries.

Blocked
This message is currently unavailable because the message point is blocked by the first message on the queue. The first message has reached its maximum failed delivery limit but no exception destination is configured. Identify the first message and resolve the problem that is preventing it from being consumed.

Transaction ID
The local transaction identifier of the transaction that this message is currently part of.

Buttons

Delete	Delete the selected items.
Delete all	Delete all items in the list.
Refresh	Refresh the number of messages.

Messages [Settings]

The properties for a message on the message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points -> *queue_point_name* -> Runtime > Messages -> *message_name*.

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Messaging engine message identifier:

Messaging engine message identifier.

Required	No
Data type	String

State:

The current state of the message related to a transaction that the message is part of. If the message is part of a transaction, the transaction identifier is shown in the Transaction ID field.

Table 217. Message transaction state.. The first column of the table lists the transaction state of the messages. The second column provides the transaction identifier if the message is part of a transaction. The third column provides the comments.

State	Transaction ID	Comments
Locked	An identifier is shown.	The message is locked as part of the transaction identified. The message has not been consumed and the transaction is still in progress.
Locked	An identifier is not shown.	The message is locked, but the transaction has completed, so the message might not have been consumed. This indicates that some error might have occurred.

Required	No
Data type	drop-down list

Range

Available
The message is available for consumption.

Locked
The message is currently unavailable. The message is in this state temporarily, possibly because it is being consumed by a non-transacted consumer.

Remote lock
The message is currently locked to a consumer attached to another, remote, messaging engine in the bus. The message will remain locked until the remote messaging engine responds with a decision on the message. If the remote messaging engine is stopped, the message will remain locked until the messaging engine is restarted. A corresponding message request for a "known remote queue point" will identify the remote messaging engine that is making the request.

Removing
The message is currently being removed under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Committing
The message is currently being added under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Pending retry
The message is currently unavailable before being eligible for a retry. This might be because a message-driven bean is configured to delay failing message retries.

Blocked
This message is currently unavailable because the message point is blocked by the first message on the queue. The first message has reached its maximum failed delivery limit but no exception destination is configured. Identify the first message and resolve the problem that is preventing it from being consumed.

Transaction ID:

The local transaction identifier of the transaction that this message is currently part of.

This field combined with the State field shows the state of the message related to the transaction. For more information about the combination of these fields, see Table 217 on page 2106.

Required No
Data type String

Run-time message properties:

Message type:

The type of the message (JMS, SDO, SUBSCRIPTION, TRM, BROKER_CONTROL, BROKER_RESPONSE, BROKER_ADMIN).

Required	No
Data type	String

Approximate length:

The approximate length of the message.

Required	No
Data type	String

Time stamp:

The time stamp of when the message was originally sent.

Required	No
Data type	String

Message wait time:

The time the message has been waiting to be consumed.

Required	No
Data type	String

Current messaging engine arrival time:

The time that the message arrived on the current messaging engine.

Required	No
Data type	String

Redelivered count:

The number of times that the message has been redelivered.

Required	No
Data type	String

Security user ID:

The security user ID.

Required	No
Data type	String

Producer type:

The producer type (API, Core, TRM).

Required	No
Data type	String

Exception destination timestamp:

The timestamp at which the message was put to the exception destination.

Required	No
Data type	String

Exception destination reason:

The reason the message was put to the exception destination.

Required	No
Data type	Text area

API Message properties:

Message ID:

The message ID.

Required	No
Data type	String

Correlation ID:

The API correlation ID for request/response correlation.

Required	No
Data type	String

User ID:

The user ID.

Required	No
Data type	String

Format:

The Format of the message.

Required	No
Data type	String

JMS Message properties:

JMS delivery mode:

The JMS delivery mode (Persistent, Non-persistent).

Required	No
Data type	String

JMS expiration:

The JMS expiration.

Required	No
Data type	String

JMS destination:

The JMS destination.

Required	No
Data type	String

JMS reply to destination:

The JMS reply to destination.

Required	No
Data type	String

JMS redelivered:

The JMS redelivered flag.

Required	No
Data type	String

JMS type:

The JMS type field (Text, Byte, Stream, Object, Map).

Required	No
Data type	String

JMSX delivery count:

The JMSX delivery count.

Required	No
Data type	String

JMSX application ID:

The JMSX application ID.

Required	No

Data type String

Bus message properties:

Topic:

The bus discriminator.

Required No
Data type String

Priority:

The bus priority (0-9).

Required No
Data type String

Reliability:

The bus reliability (Assured persistent, Reliable persistent, Reliable non-persistent, Express non-persistent, Best effort non-persistent).

Required No
Data type String

Time to live:

The time to live of the message.

Required No
Data type String

Reply discriminator:

The reply discriminator value of the bus.

Required No
Data type String

Reply priority:

The reply priority value of the bus (0-9).

Required No
Data type String

Reply reliability:

The reply reliability value of the bus (Assured persistent, Reliable persistent, Reliable non-persistent, Express non-persistent, Best effort non-persistent).

Required No
Data type String

Reply time to live:

The reply time to live value of the bus.

Required No
Data type String

System message ID:

The unique id of the message, assigned by the bus.

Required No
Data type String

Additional Properties

Message body
The contents of the message body.

Messages [Settings]

The properties for a message on the message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points -> *queue_point_name* -> Runtime > **Messages -> *message_name***.

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Messaging engine message identifier:

Messaging engine message identifier.

Required No
Data type String

State:

The current state of the message related to a transaction that the message is part of. If the message is part of a transaction, the transaction identifier is shown in the Transaction ID field.

Required No

Data type drop-down list

Range

Available
The message is available for consumption.

Locked
The message is currently unavailable. The message is in this state temporarily, possibly because it is being consumed by a non-transacted consumer.

Remote lock
The message is currently locked to a consumer attached to another, remote, messaging engine in the bus. The message will remain locked until the remote messaging engine responds with a decision on the message. If the remote messaging engine is stopped, the message will remain locked until the messaging engine is restarted. A corresponding message request for a "known remote queue point" will identify the remote messaging engine that is making the request.

Removing
The message is currently being removed under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Committing
The message is currently being added under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Pending retry
The message is currently unavailable before being eligible for a retry. This might be because a message-driven bean is configured to delay failing message retries.

Blocked
This message is currently unavailable because the message point is blocked by the first message on the queue. The first message has reached its maximum failed delivery limit but no exception destination is configured. Identify the first message and resolve the problem that is preventing it from being consumed.

Transaction ID:

The local transaction identifier of the transaction that this message is currently part of.

Required No
Data type String

Run-time message properties:

Message type:

The type of the message (JMS, SDO, SUBSCRIPTION, TRM, BROKER_CONTROL, BROKER_RESPONSE, BROKER_ADMIN).

Required	No
Data type	String

Approximate length:

The approximate length of the message.

Required	No
Data type	String

Time stamp:

The time stamp of when the message was originally sent.

Required	No
Data type	String

Message wait time:

The time the message has been waiting to be consumed.

Required	No
Data type	String

Current messaging engine arrival time:

The time that the message arrived on the current messaging engine.

Required	No
Data type	String

Redelivered count:

The number of times that the message has been redelivered.

Required	No
Data type	String

Security user ID:

The security user ID.

Required	No
Data type	String

Producer type:

The producer type (API, Core, TRM).

Required	No
Data type	String

Exception destination timestamp:

The timestamp at which the message was put to the exception destination.

Required	No
Data type	String

Exception destination reason:

The reason the message was put to the exception destination.

Required	No
Data type	Text area

API Message properties:

Message ID:

The message ID.

Required	No
Data type	String

Correlation ID:

The API correlation ID for request/response correlation.

Required	No
Data type	String

User ID:

The user ID.

Required	No
Data type	String

Format:

The Format of the message.

Required	No
Data type	String

JMS Message properties:

JMS delivery mode:

The JMS delivery mode (Persistent, Non-persistent).

Required	No
Data type	String

JMS expiration:

The JMS expiration.

Required	No
Data type	String

JMS destination:

The JMS destination.

Required	No
Data type	String

JMS reply to destination:

The JMS reply to destination.

Required	No
Data type	String

JMS redelivered:

The JMS redelivered flag.

Required	No
Data type	String

JMS type:

The JMS type field (Text, Byte, Stream, Object, Map).

Required	No
Data type	String

JMSX delivery count:

The JMSX delivery count.

Required	No
Data type	String

JMSX application ID:

The JMSX application ID.

Required	No

Data type String

Bus message properties:

Topic:

The bus discriminator.

Required No
Data type String

Priority:

The bus priority (0-9).

Required No
Data type String

Reliability:

The bus reliability (Assured persistent, Reliable persistent, Reliable non-persistent, Express non-persistent, Best effort non-persistent).

Required No
Data type String

Time to live:

The time to live of the message.

Required No
Data type String

Reply discriminator:

The reply discriminator value of the bus.

Required No
Data type String

Reply priority:

The reply priority value of the bus (0-9).

Required No
Data type String

Reply reliability:

The reply reliability value of the bus (Assured persistent, Reliable persistent, Reliable non-persistent, Express non-persistent, Best effort non-persistent).

Required No
Data type String

Reply time to live:

The reply time to live value of the bus.

Required No
Data type String

System message ID:

The unique id of the message, assigned by the bus.

Required No
Data type String

Additional Properties

Message body
The contents of the message body.

Messages [Settings]

The properties for a message on the message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points -> *queue_point_name* -> Runtime > **Messages -> *message_name***.

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Messaging engine message identifier:

Messaging engine message identifier.

Required No
Data type String

State:

The current state of this message on the message point. If the message is part of a transaction, the transaction identifier is shown in the Transaction ID field.

Required No

Data type drop-down list

Range

Available
The message is available for consumption.

Locked
The message is currently unavailable. The message is in this state temporarily, possibly because it is being consumed by a non-transacted consumer.

Remote lock
The message is currently locked to a consumer attached to another, remote, messaging engine in the bus. The message will remain locked until the remote messaging engine responds with a decision on the message. If the remote messaging engine is stopped, the message will remain locked until the messaging engine is restarted. A corresponding message request for a "known remote queue point" will identify the remote messaging engine that is making the request.

Removing
The message is currently being removed under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Committing
The message is currently being added under a transaction. The message will be in this state until the transaction commits or rolls back. If this state persists, investigate the state of the transaction identified by **Transaction ID**.

Pending retry
The message is currently unavailable before being eligible for a retry. This might be because a message-driven bean is configured to delay failing message retries.

Blocked
This message is currently unavailable because the message point is blocked by the first message on the queue. The first message has reached its maximum failed delivery limit but no exception destination is configured. Identify the first message and resolve the problem that is preventing it from being consumed.

Transaction ID:

The local transaction identifier of the transaction that this message is currently part of.

Required No
Data type String

Run-time message properties:

Message type:

The type of the message (JMS, SDO, SUBSCRIPTION, TRM, BROKER_CONTROL, BROKER_RESPONSE, BROKER_ADMIN).

Required	No
Data type	String

Approximate length:

The approximate length of the message.

Required	No
Data type	String

Time stamp:

The time stamp of when the message was originally sent.

Required	No
Data type	String

Message wait time:

The time the message has been waiting to be consumed.

Required	No
Data type	String

Current messaging engine arrival time:

The time that the message arrived on the current messaging engine.

Required	No
Data type	String

Redelivered count:

The number of times that the message has been redelivered.

Required	No
Data type	String

Security user ID:

The security user ID.

Required	No
Data type	String

Producer type:

The producer type (API, Core, TRM).

Required	No
Data type	String

Exception destination timestamp:

The timestamp at which the message was put to the exception destination.

Required	No
Data type	String

Exception destination reason:

The reason the message was put to the exception destination.

Required	No
Data type	Text area

API Message properties:

Message ID:

The message ID.

Required	No
Data type	String

Correlation ID:

The API correlation ID for request/response correlation.

Required	No
Data type	String

User ID:

The user ID.

Required	No
Data type	String

Format:

The Format of the message.

Required	No
Data type	String

Bus message properties:

Topic:

The bus discriminator.

Required	No
Data type	String

Priority:

The bus priority (0-9).

Required	No
Data type	String

Reliability:

The bus reliability (Assured persistent, Reliable persistent, Reliable non-persistent, Express non-persistent, Best effort non-persistent).

Required	No
Data type	String

Time to live:

The time to live of the message.

Required	No
Data type	String

Reply discriminator:

The reply discriminator value of the bus.

Required	No
Data type	String

Reply priority:

The reply priority value of the bus (0-9).

Required	No
Data type	String

Reply reliability:

The reply reliability value of the bus (Assured persistent, Reliable persistent, Reliable non-persistent, Express non-persistent, Best effort non-persistent).

Required	No
Data type	String

Reply time to live:

The reply time to live value of the bus.

Required	No
Data type	String

System message ID:

The unique id of the message, assigned by the bus.

Required	No
Data type	String

Additional Properties

Message body
The contents of the message body.

Messaging engines [Collection]

A messaging engine is a component, running inside a server, that manages messaging resources for a bus member. Applications are connected to a messaging engine when they access a service integration bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the messaging engine.

Description
An optional description for the messaging engine, for administrative purposes.

Status
The current status of the messaging engine. Status settings are started or stopped.

Buttons

Enable policy assistance	This button only applies to messaging engines in a clustered environment. Start the wizard that provides messaging engine policy assistance, that is, guidance about creating one or more messaging engines for the cluster and configuring the messaging engine behavior. This button is available only when messaging engine policy assistance is not enabled already.
Add messaging engine	This button only applies to messaging engines in a clustered environment. Add a new messaging engine to the server cluster.

Remove messaging engine	This button only applies to messaging engines in a clustered environment. Remove the selected messaging engines from the server cluster.
Start	Start selected items.
Stop	Stop the selected messaging engines. You must first select the messaging engines to be stopped. You can select a Stop mode: **Immediate** The messaging engine stops after all messaging operations that are occurring at the time of the stop request have completed. After a stop request is issued, no new messaging operations can start. For each existing connection, the messaging engine waits for the current operation to complete, unless the operation is one that blocks processing in the messaging engine, such as a receive operation. In this case, the operation is interrupted. Asynchronous consumers can complete, even though they might take an arbitrary amount of time to process the current message. The messaging engine then backs out of active transactions and disallows any further operations on that connection. When all connections are in this invalidated state, the messaging engine stops. **Force** The messaging engine stops without allowing messaging operations to complete. Applications are forcefully disconnected. This mode completes the shutdown of the messaging engine in as short a time as possible. When a messaging engine that was stopped by using force mode is restarted, the restart might take longer than if it was stopped using immediate mode, because more recovery actions are needed. For example, messages might be left in an indoubt state and you must deal with these messages to resolve any indoubt transactions. **Tip:** If an immediate stop takes too long, you can select the Force option to escalate it to a force stop. This overrides your previous selection of the Immediate option.

Messaging engines [Settings]

A messaging engine is a component, running inside a server, that manages messaging resources for a bus member. Applications are connected to a messaging engine when they access a service integration bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name*.

- “Configuration tab”
- “Runtime tab” on page 2128

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the messaging engine.

Required No
Data type String

UUID:

The universal unique identifier assigned by the system to this messaging engine for administrative purposes.

Required No
Data type String

Description:

An optional description for the messaging engine, for administrative purposes.

Required No
Data type Text area

Initial state:

The initial state determines whether the messaging engine is started automatically.

If this property is selected, the messaging engine starts automatically when its associated application server is started. Until the messaging engine starts, it is unavailable.

Required No
Data type drop-down list
Range

Stopped
When the associated application server is started, the messaging engine is stopped and is not available to process messages.

Started
When the associated application server is started, the messaging engine is started and is available to process messages.

Message store type:

The type of message store used. Either file store or data store. Once the messaging engine has been created this cannot be changed, only configured.

Required No
Data type drop-down list
Range

File store
A file store is a type of message store that directly uses files in a file system through the operating system.

Data store
A data store consists of the set of tables that a messaging engine uses to store persistent data in a database.

High message threshold per message point:

The number of messages queued on a message point on this messaging engine, at which point new messages are not accepted on the message point. However, certain messages that are already in the bus and that are being transmitted to this messaging engine might be accepted.

When the messaging engine is created, the high message threshold of the bus is used to set the default value for this property. When a message point is created on this messaging engine, the value of this property is used to set the default high message threshold for the message point.

Required No
Data type Integer
Range 1 through 9223372036854775807

Note: There are implications for the Java Virtual Machine (JVM) heap size when using a high message threshold, if large numbers of messages are held on a queue (each message consumes approximately 200 bytes of storage). Therefore if you increase the high message threshold because you are expecting large numbers of messages on your queues, you should also modify the JVM heap size of the server as appropriate.

Default blocked destination retry interval:

The time delay, in milliseconds, that is introduced by the system under certain circumstances before a failed message delivery to an application will be retried. This delay can be overridden by individual Queue destination configurations.

When you configure a bus destination, you can specify an associated exception destination and a maximum number of times that an individual message fails to be consumed before it is put on that exception destination. Alternatively, if you do not specify an associated exception destination, the system continues to try to deliver the message. In this situation, the system attempts to deliver the message, without applying any delay, until it reaches the maximum failed deliveries limit set for the bus destination (a queue or a topic space). After the maximum failed deliveries limit is reached, the default blocked destination retry interval is applied before the message is retried.

The Default blocked destination retry interval specifies the time interval between retry attempts, which will be used by all queue and topic destinations that are associated with this messaging engine. You can override this default value when you configure an individual queue or topic destination.

Required	No
Data type	Integer (milliseconds)
Default	The value taken from the sib.processor.blockedRetryTimeout custom property, if set. Otherwise it is set to 5000.
Range	1 through 9223372036854775807

Target groups:

A list of names of target groups with which the messaging engine will register.

Custom target groups are a type of target group used by JMS connection factories. When an application creates a connection to a service integration bus, it uses connection factory properties to specify suitable messaging engines to connect to. When a target type of “Custom” is specified in the connection factory **targetType** property, the application is connected to one of the messaging engines in the specified custom target group. A particular messaging engine is selected from the group according to the other connection factory properties that are specified.

Required	No
Data type	Text area

Bus name:

The name of the service integration bus on which the messaging engine is configured.

Required	No
Data type	String

Bus UUID:

The universal unique identifier of the service integration bus on which the messaging engine is configured.

Required	No
Data type	String

Message points

Mediation points

A mediation point is created on each messaging engine to which the mediation of a mediated destination is assigned. It is used to hold messages pending delivery to the mediation.

Queue points

A queue point is created on each messaging engine to which a point-to-point destination is assigned. It is used to hold messages pending delivery to receiving applications.

Publication points

A publication point is created on each messaging engine in the bus when a publish/subscribe destination is created. The publication point on a messaging engine is used to hold messages published by applications connected to that messaging engine until they are delivered to subscribers.

Additional Properties

Mediation execution points
Mediation execution points for the processing of messages from mediation message points that are on a WebSphere MQ server.

Service integration bus links
A communications link between this messaging engine and a messaging engine in a foreign service integration bus.

Mediation thread pool
The thread pool for allocating mediation handler threads.

Message store
The properties for the message store in use by this messaging engine as determined by the message store type field. The message store can be either a file store or a data store.

WebSphere MQ client links
A WebSphere MQ client link enables JMS client applications to connect, via this messaging engine, to the service integration bus, as though it were a WebSphere MQ queue manager.

WebSphere MQ links
Links between the messaging engine and WebSphere MQ networks. Each WebSphere MQ link connects the messaging engine as a queue manager to WebSphere MQ, providing a bridge between the bus and a WebSphere MQ network.

Custom properties
Arbitrary name-value pairs of data, where the name is a property key and the value is a string value that can be used to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console.

Reliable messaging state
Use this page to view and manage the WS-ReliableMessaging runtime state.

Related Items

Bus member
The bus member associated with this messaging engine.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Status:

The current status of the messaging engine. Status settings are started or stopped.

To be able to retrieve the status of messaging engines, you must be logged into the administrative console with at least monitor authority. If you do not have this authority, the messaging engine status is displayed as "Unavailable", even if the messaging engine has started.

Required	No
Data type	String

Message points

Queue points
A queue point is created on each messaging engine to which a point-to-point destination is assigned. It is used to hold messages pending delivery to receiving applications.

Publication points
A publication point is created on each messaging engine in the bus when a publish/subscribe destination is created. The publication point on a messaging engine is used to hold messages published by applications connected to that messaging engine until they are delivered to subscribers.

Mediation points
A mediation point is created on each messaging engine to which the mediation of a mediated destination is assigned. It is used to hold messages pending delivery to the mediation.

Remote message points

Remote queue points
The remote queue points that are producing/consuming messages to/from queue points on remote messaging engines.

Remote mediation points
The remote mediation points that are producing messages for mediation points on remote messaging engines.

Remote publication points
The remote publication points that are producing messages for publication points on remote messaging engines.

Messaging resources for this application

This pane provides an expandable tree view of all the references to messaging resources declared in the deployment descriptors for the current application. As many of the references as possible are resolved to links to the associated configuration panel for the referenced object.

To view this page in the console, click the following path:

Applications -> Application Types -> WebSphere enterprise applications -> *application_name* -> [Enterprise Java Bean Properties] Default messaging provider references.

Use this panel to inspect the configuration from the application to the queue (destination) to ensure that the configuration is correct.

Using the default messaging provider, a JMS activation specification refers to a JMS destination (either a JMS Queue or JMS Topic) that in turn points to a service integration bus destination (either a queue destination or a topic space destination). JMS resources are referenced through Java Naming and Directory Interface (JNDI) names and service integration resources are referenced through bus and resource identifiers. This panel enables you to review the dependencies within your configuration, making it easier to detect inconsistencies, for example as a result of references being entered incorrectly, or JNDI or resource names being changed or deleted without the associated configuration having been updated.

Problems with the configuration are usually detected in one of two ways:

- An application can no longer send or receive messages.
- A destination becomes full and can no longer receive messages because the existing messages are not being consumed.

This panel can help you find the cause of the problem by giving you a high level view of many relevant resources.

Note: For a related view of the applications and JMS resources for a given destination, see the following panel: “Application resources for this destination” on page 2035.

- “Local Topology tab”

Local Topology tab

Topology properties for this object. These properties detail how this object relates to other objects in the system topology.

Permitted transports [Collection]

A permitted transport is a transport mechanism that this bus will allow remote clients to use.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Additional Properties] Permitted transports.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Transport Name
The name of the permitted transport.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Port [Settings]

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *port_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The identifier by which this destination is known for administrative purposes.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this destination for administrative purposes.

Required	No
Data type	String

Type:

Whether this bus destination is for a queue, topic space, or some other type of destination.

Required	No
Data type	String

Description:

An optional description for the bus destination, for administrative purposes.

Required	No
Data type	Text area

Mediation:

The name of the mediation that mediates this destination.

Required	No
Data type	String

Default reliability:

The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

Required	No
Data type	drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Maximum reliability:

The maximum reliability of messages accepted by this destination.

Required No
Data type drop-down list
Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Enable producers to override default reliability:

Select this option to enable producers to override the default reliability that is set on the destination.

Required	No
Data type	Boolean

Default priority:

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

Required	No
Data type	Custom

Exception destination:

Use these properties to define what happens to any messages that cannot be delivered to this destination.

Use this property to define what happens to any messages that cannot be delivered to this destination.

By default, such messages are routed to the system default exception destination of the messaging engine that discovers the problem: _SYSTEM.Exception.Destination.*engine_name*.

If you want messages to be sent to another exception destination, select Specify then type the exception destination name. The exception destination must be a queue, on the same bus or a foreign bus, and must exist when the destination is created.

If you do not want undeliverable messages to be sent to an exception destination, select None.

Required	No
Data type	String and Boolean

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

Required	No
Data type	Boolean

Receive allowed:

Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination.

Required	No
Data type	Boolean

Receive exclusive:

Select this option to allow only one consumer to attach to each message point. If this option is not selected multiple consumers will be allowed to attach and receive messages from each message point.

Required	No
Data type	Boolean

Maintain strict message order:

Enabling this option will maintain the strict ordering of messages for this destination.

Required	No
Data type	Custom

Additional Properties

Context properties
Context information passed to the mediation.

Message points

Queue points
A queue point is created on each messaging engine to which a point-to-point destination is assigned. It is used to hold messages pending delivery to receiving applications.

Mediation points
A mediation point is a location in a messaging engine at which messages are mediated. A mediation point is created when a mediation is associated with a bus destination.

Property [Settings]

Use this page to specify an arbitrary name and value pair. The value that is specified for the name and value pair is a string that can set internal system configuration properties.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Additional Properties] Custom properties -> *property_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

Specifies the name of the property.

Required	Yes
Data type	String

Value:

Specifies the value that is paired with the specified name.

Required	Yes
Data type	String

Description:

Specifies a description of the name and value pair. The description should help to differentiate this pair for other defined pairs.

Required	No
Data type	Text area

Optional:

Specifies an optional attribute that determines whether this property must have a value.

Required	No
Data type	Boolean

Validation Expression:

Specifies a value that the administrative console and some host tools use to validate the contents of the value of this property.

Required	No
Data type	String

Publication points [Collection]

The message point for a topic space, for publish/subscribe messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name* -> [Message points] Publication points.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Queue Depth
The total set of messages currently queued by the publication point for the attached subscriptions.

These messages can be:

- Shared across multiple local subscriptions
- Queued for a single subscription
- Queued for transmission to one or more remote publication points

Investigate the reason for queued messages, as follows:

- Display the runtime collection of subscriptions for this publication point and identify the subscriptions with non-zero queue depths.

- Also display the Remote publication point collection panel for this messaging engine and identify any remote publication points for the same topic space as this publication point that have a non-zero current outbound message value. If these exist, ensure that the target messaging engines are started.

Publication points [Collection]

The message point for a topic space, for publish/subscribe messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name* -> [Message points] Publication points -> Runtime.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Publication points [Settings]

The message point for a topic space, for publish/subscribe messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name* -> [Message points] Publication points -> *publication_point_name*.

- "Configuration tab"
- "Runtime tab" on page 2137

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this message point for administrative purposes.

Required	No

Data type String

Destination type:

Whether the message point is a queue or topic space.

Required No
Data type String

High message threshold:

A threshold above which the messaging system will take action to limit the addition of more messages to this message point.

A publication point stores a single copy of each message that is currently queued to a subscription on this publication point. Multiple subscriptions that are queueing the same message will result in a single copy of the message on the publication point. A publication point can reach its high message threshold because a single subscription has many messages queued to it or because many subscriptions have a few, different messages queued to them. If the high message threshold is reached, investigate the state of the individual subscriptions on this publication point to identify the cause.

Required No
Data type Long
Range 1 through 9223372036854775807

Send allowed:

Clear this option (setting it to false) to stop messages from being put onto this message point. This value will be overridden by the parent destination if that destination has sendAllowed disabled, which stops messages from being put onto all its message points.

Required No
Data type Boolean

Target UUID:

The UUID of the bus destination for which this is a message point.

Required No
Data type String

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required	No
Data type	String

Run-time ID:

The internal runtime identifier assigned to this message point.

Required	No
Data type	String

High message threshold:

A threshold above which the messaging system will take action to limit the addition of more messages to this message point.

Required	No
Data type	String

Send allowed:

Clear this option (setting it to false) to stop messages from being put onto this message point. This value will be overridden by the parent destination if that destination has sendAllowed disabled, which stops messages from being put onto all its message points.

Required	No
Data type	Boolean

Additional Properties

Subscriptions
The active subscriptions for the topic space.

Known remote publication points
The remote messaging engines that have remote producers connected to this publication point.

Publish/subscribe broker profiles [Collection]

Profiles used to define the topic mappings and transactionality for publishing and receiving (by subscription) topics across the publish/subscribe bridge between WebSphere Application Server and a message broker in a WebSphere MQ network.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the publish/subscribe broker profile.

Description
An optional description for the publish/subscribe broker profile, for administrative purposes.

Broker queue manager
The name of the queue manager for the WebSphere MQ broker.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Publish/subscribe broker profiles [Settings]

Profiles used to define the topic mappings and transactionality for publishing and receiving (by subscription) topics across the publish/subscribe bridge between WebSphere Application Server and a message broker in a WebSphere MQ network.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name*.

- "Runtime tab"
- "Configuration tab" on page 2140

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Number of subscriptions:

The current number of broker subscriptions for this broker profile.

Required No
Data type String

Additional Properties

Subscriptions
The list of current broker subscriptions for this broker profile.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the publish/subscribe broker profile.

This name is only for administrative purposes. If you created this broker profile by using the foreign bus connection wizard, then this name is generated automatically by adding "_broker_profile" to the end of the broker queue manager name.

Required	Yes
Data type	String

Description:

An optional description for the publish/subscribe broker profile, for administrative purposes.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Text area

Broker queue manager:

The name of the queue manager for the WebSphere MQ broker.

This identifies the queue manager to which the message broker is connected in the WebSphere MQ network. It does not have to be the WebSphere MQ gateway queue manager.

Required	No
Data type	String

Additional Properties

Topic mappings

Topic mappings for publishing and receiving (by subscription) messages across the publish/subscribe bridge between WebSphere Application Server and a message broker in a WebSphere MQ network.

Queue points [Collection]

The message point for a queue, for point-to-point messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Queue points [Collection]

The message point for a queue, for point-to-point messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points -> Runtime.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Queue points [Settings]

The message point for a queue, for point-to-point messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *queue_name* -> [Message points] Queue points -> *queue_point_name*.

- "Configuration tab"
- "Runtime tab" on page 2142

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Buttons

Refresh	Refresh the number of messages.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this message point for administrative purposes.

Required	No
Data type	String

Destination type:

Whether the message point is a queue or topic space.

Required	No
Data type	String

High message threshold:

A threshold above which the messaging system will take action to limit the addition of more messages to this message point.

Required	No
Data type	Long
Range	1 through 9223372036854775807

Send allowed:

Clear this option (setting it to false) to stop messages from being put onto this message point. This value will be overridden by the parent destination if that destination has sendAllowed disabled, which stops messages from being put onto all its message points.

Required	No
Data type	Boolean

Target UUID:

The UUID of the bus destination for which this is a message point.

Required	No
Data type	String

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Buttons

Refresh	Refresh the number of messages.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required No
Data type String

Run-time ID:

The internal runtime identifier assigned to this message point.

Required No
Data type String

High message threshold:

A threshold above which the messaging system will take action to limit the addition of more messages to this message point.

Required No
Data type String

Send allowed:

Clear this option (setting it to false) to stop messages from being put onto this message point. This value will be overridden by the parent destination if that destination has sendAllowed disabled, which stops messages from being put onto all its message points.

Required No
Data type Boolean

Current message depth:

The number of messages on the message point.

Required No
Data type String

Additional Properties

Messages
Messages queued on the queue point.

Known remote queue points
The remote messaging engines that have remote producers or consumers connected to this queue point.

Remote mediation points [Collection]

The remote mediation points that are producing messages to mediation points on remote messaging engines.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote mediation points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A remote mediation point represents a remote proxy for a mediation point that exists at a remote messaging engine to which messages are sent by an application or client connected to this messaging engine. The remote mediation point manages an outbound message stream to the mediation point at the remote messaging engine.

Identifier
The remote mediation point identifier.

Messaging engine
The remote messaging engine where this mediation point is localized.

Current outbound messages
The current number of outbound messages queued to the mediation point.

Outbound messages sent
The total number of messages sent to the mediation point since the messaging engine started.

Buttons

Delete all messages	Delete all messages from the outbound message stream of the selected remote queue point. Messages in “Pending Acknowledgment” state might already have been received at the remote queue point. You cannot delete messages in “Committing” state; you must first commit their transactions.
Move all messages	Move all messages from the outbound message stream of the selected remote queue point to the exception destination configured for the destination. Messages in “Pending Acknowledgment” state might already have been received at the remote queue point. Moving these messages to the exception destination results in two copies of the message in the bus. You cannot move messages in “Committing” state; you must first commit their transactions.

Remote mediation points [Settings]

The remote mediation points that are producing messages to mediation points on remote messaging engines.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote mediation points** -> *identifier_name* .

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The remote mediation point identifier.

Required	No
Data type	String

Messaging engine:

The remote messaging engine where this mediation point is localized.

Required	No
Data type	String

Current outbound messages:

The current number of outbound messages queued to the mediation point.

Required	No
Data type	String

Outbound messages sent:

The total number of messages sent to the mediation point since the messaging engine started.

Required	No
Data type	String

Additional Properties

Outbound messages

The outbound message streams from this remote message point to the remote localized message point.

Outbound messages [Collection]

The outbound message streams from this remote message point to the remote localized message point.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote queue points -> *identifier_name* -> [Additional Properties] Outbound messages**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

The collection of outbound message streams ordered by quality of service (priority and reliability) for messages sent to a message point on a remote messaging engine.

Priority
The stream priority.

Reliability
The stream reliability.

Number of messages
The current number of messages on the stream.

Status
The status of the stream.

Messages
View the messages on the selected stream.

Remote Publication Points [Collection]

The remote publication points that are producing messages to publication points on remote messaging engines.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote publication points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A remote publication point represents a remote proxy for a publication point that exists at a remote messaging engine, for which publications are published to by an application or client connected to this messaging engine. The remote publication point manages an outbound message stream to the publication point at the remote messaging engine

Identifier
The remote publication point identifier.

Message engine
The remote messaging engine where this publication point is localized.

Current outbound messages
The current number of outbound messages queued to the publication point.

Outbound messages sent
The total number of messages sent to the publication point since the messaging engine started.

Buttons

Delete all messages	Delete all messages from the outbound message stream of the selected remote queue point. Messages in "Pending Acknowledgment" state might already have been received at the remote queue point. You cannot delete messages in "Committing" state; you must first commit their transactions.

Remote Publication Points [Settings]

The remote publication points that are producing messages to publication points on remote messaging engines.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote publication points -> *identifier_name*** .

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The remote publication point identifier.

Required No
Data type String

Message engine:

The remote messaging engine where this publication point is localized.

Required No
Data type Text area

Current outbound messages:

The current number of outbound messages queued to the publication point.

Required	No
Data type	String

Outbound messages sent:

The total number of messages sent to the publication point since the messaging engine started.

Required	No
Data type	String

Outbound properties

Messages
The messages queued outbound to the publication point on the remote messaging engine.

Topics
The topics that have been subscribed to from this remote messaging engine.

Inbound properties

Remote subscriptions
The subscriptions that have been made from this messaging engine to subscription homes on remote messaging engines.

Remote subscription [Collection]

The subscription that has been made from this messaging engine to a subscription home on a remote messaging engine.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote Message Points] Remote publication points** > ***identifier_name*** **-> [Inbound Properties] Remote subscriptions**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

The subscriptions that have been made by subscribing applications, or clients connected to this messaging engine, to the remote messaging engine where the subscriptions' "Homes" exist.

Name The name of the remote subscription.

Current message requests
The current number of active message retrieval requests sent to the publication point from this remote subscription point.

Completed message requests
The total number of completed message retrieval requests sent to the publication point from this remote subscription point since the messaging engine started.

Remote subscription [Settings]

The subscription that has been made from this messaging engine to a subscription home on a remote messaging engine.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote Message Points] Remote publication points** > ***identifier_name*** **-> [Inbound Properties] Remote subscriptions -> *subscription_name*** .

- “Runtime tab”

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the remote subscription.

Required	No
Data type	String

Current message requests:

The current number of active message retrieval requests sent to the publication point from this remote subscription point.

Required	No
Data type	String

Completed message requests:

The total number of completed message retrieval requests sent to the publication point from this remote subscription point since the messaging engine started.

Required	No
Data type	String

Additional Properties

Message requests

The total number of active and completed message retrieval requests sent to the publication point from this remote subscription point since the messaging engine started.

WebSphere MQ sender channel saved batch status [Collection]

The saved status of message batches for the sender channel saved for transmission to WebSphere MQ. You can choose to commit or rollback each batch.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Sender channel -> *channel_name* -> Runtime > **Saved batch status**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Virtual queue manager name
The virtual queue manager name by which the local bus is known to a WebSphere MQ network.

MQ channel name
The name of the channel used for the WebSphere MQ link sender channel.

In doubt
Whether or not the saved batch is in doubt.

Current LUWID
The current logical unit-of-work identifier for the saved batch.

Current sequence number
The current sequence number for the saved batch.

Last LUWID
The last logical unit-of-work identifier for the saved batch.

Last sequence number
The last sequence number for the saved batch.

Buttons

Commit	Commit the logical unit-of-work for the message batch.
Roll back	Roll back the logical unit-of-work for the message batch.

WebSphere MQ receiver channel saved batch status [Collection]

The runtime status of message batches for the receiver channel of the WebSphere MQ link.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Receiver channel -> *channel_name* -> Runtime > **Saved batch status**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

WebSphere MQ Queue Manager name
The name of the WebSphere MQ queue manager from which the message batch was received.

MQ channel name
The name of the WebSphere MQ sender channel from which the message batch was received.

Last LUWID
The last unit of work identifier for the message batch.

Last sequence number
The last sequence number for the message batch.

Security domain configuration. [Settings]

Configure the security settings for the security domain to which your service integration bus is assigned.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Bus security domain] Configure security domain.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

Use this pane to create a customized security domain for the selected bus. By having a customized security domain for each bus in your bus topology, you can configure different security attributes for each bus within a cell. You can view the current security settings for the security domain to which the selected service integration bus is assigned, and to make changes to the existing domain configuration.

• “Configuration tab”

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Domain name:

The unique name for this security domain.

You must type a unique name for the security domain.

Required	No
Data type	String

Domain description:

A description of this security domain.

Required	No
Data type	String

User Realm::

Specifies the settings for the user realm.

Click the **Expand** icon to display the options for configuring the user realm.

You can use existing global security settings, or customize a user realm specifically for this domain. If you customize a user realm, you must select the type of realm, and you might have to configure the user registry.

Required	No
Data type	Expand

JAAS System Logins::

Specifies the settings for the JAAS system logins.

Click the **Expand** icon to display the options for configuring JAAS system logins.

Required	No
Data type	Expand

JAAS J2C Authentication Data::

Specifies the settings for the JAAS J2C authentication data.

Click the **Expand** icon to display the options for configuring authentication data for JAAS Java Platform Enterprise Edition (Java EE) Connector Architecture authentication data.

Required	No
Data type	Expand

Security for bus *bus_name* [Settings]

Configure the security settings for your service integration bus.

To view this pane in the console, click one of the following paths:

Service integration -> Buses -> *bus_name* > **[Additional Properties] Security**.

Service integration -> Buses -> *security_value*.

The value of *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Launch Bus Security Wizard

Click to start a wizard to configure the security settings bus for a bus. If the wizard detects that bus security is disabled, you are prompted to enable it.

General Properties

Enable bus security:

Select this option to inherit the secure administration setting of the cell. Deselect this option if you always wish to disable bus security.

Required	No
Data type	Boolean

Inter-engine authentication alias:

The name of the authentication alias used to authorize communication between messaging engines on the bus.

You must specify an inter-engine authentication alias if the bus contains a WebSphere Application Server Version 6 bus member. When bus security is enabled, the bus uses the inter-engine authentication alias to authenticate incoming connections from other messaging engines. An unauthorized messaging engine cannot connect to the bus.

Required	No
Data type	drop-down list

Permitted transports:

Select the type of allowed permitted transports.

Allow the use of all defined transport channel chains

Restrict the use of defined transport channel chains to those protected by SSL

Restrict the use of defined transport channel chains to the list of permitted transports

To ensure that all ports used by the bus are secure, select **Restrict the use of defined transport channel chains to those protected by SSL** , or if your permitted transport chains are secure, select **Restrict the use of defined transport channel chains to the list of permitted transports**. This prevents the InboundBasicMessaging port being opened. Changes to this setting are effective when the server is restarted.

Required	No
Data type	Radio button

Use the Server ID when running mediations:

Check this option if you want to run mediations using the server identity, instead of using a mediation authentication alias.

Select this option if you want to run mediations on multiple servers in different domains. Using the server identity enables you to run mediations successfully across multiple security domains without having to specify a mediation authentication alias for each domain. You can also use this option when multiple domains are not in use.

Required No
Data type Boolean

Mediations authentication alias:

The name of the authentication alias used to authorize mediations to access the bus.

You must specify a mediation authentication alias if the bus contains a WebSphere Application Server Version 6 bus member. The mediations authentication alias ensures that the bus operates securely. If a mediation authentication alias is specified for a bus that contains no Version 6 bus members, it is ignored.

Required No
Data type drop-down list

Bus security domain:

Select one of the following options to assign the bus to a security domain:

Use the global security domain
Select this option to assign the bus to the global security domain. If you have a mixed-version bus, you must assign it to the global security domain.

Required No
Data type Radio button

Inherit the cell level security domain
Select this option to let the bus inherit the cell level security domain. If no cell level domain is specified then the global security domain will be used.

Required No
Data type Radio button

Use the selected domain
Select a custom security domain for this bus. This domain will be used for authentication and determining other security information.

Required No
Data type Radio button

Configure Security Domain...
Select this link to configure security settings for a custom security domain. This link becomes active only after you have applied or saved the option to use a non-global domain.

Performance:

Group cache timeout:

The length of time, in minutes, that a security group will be cached for.

Increasing the timeout decreases the load on the user registry and improves performance but makes the system less responsive to changes in a user's group membership. To tune the user's group cache to the optimum setting, you have to balance the need for responsiveness with the registry load. The default value is 120 minutes. If the system must respond quickly to changes in a user's group membership, specify a timeout of approximately 15 minutes. If it is acceptable to update a user's group membership only once a

day, for example, as an overnight process, specify a timeout of 1440 minutes (24 hours). With a setting of 0, entries in the cache do not timeout, and so remain until the server is next restarted.

A change to this value is effective immediately and only affects the group cache of the bus for which the configuration was changed.

Required	No
Data type	Long
Range	0 through 99999

Audit:

Enable the auditing service for this bus:

Required	No
Data type	Boolean

Authorization Policy

Users and groups in the bus connector role
The list of users and groups in the bus connector role.

Manage default access roles
Manage the assignment of default role types to users and groups

Manage destination access roles
Manage the assignment of destination role types to users and groups

Manage foreign bus access roles
Manage the assignment of foreign bus role types to users and groups

Manage temporary destination prefix access roles
Manage the assignment of temporary destination prefix role types to users and groups

Manage topic access roles
Manage the assignment of topic role types to users and groups

Manage users and groups not known to the user repository
Manage users and groups not known to the user repository

Additional Properties

Permitted transports
The list of permitted transports.

Related Items

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

Secure Administration and Applications
Link to configure WebSphere global security settings.

Security domains
Security domain configuration.

Audit Service
Configure the global audit settings

Sender channel transmitters [Collection]

This pane displays the transmission queue for a WebSphere MQ link sender channel.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel transmitters**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel transmitters**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

WebSphere MQ link sender channel
The name of the WebSphere MQ link sender channel for which this is the channel transmitter. Select this link to view the message collection panel for the messages on the sender channel transmitter.

Status
The runtime status of the WebSphere MQ link sender channel transmitter. Green (running), Amber (running but a problem exists), Red (stopped).

Current outbound messages
The current number of messages on the sender channel transmitter.

Outbound messages sent
The current number of messages on the sender channel transmitter.

Time since last message sent
The amount of time since the last message was sent across the sender channel or since the messaging engine started.

Known link transmitters
Select this link to view the known link transmitters for the sender channel transmitter.

Buttons

Move all messages	Move all messages on the selected WebSphere MQ link sender channel to the link exception destination.
Delete all messages	Delete all messages on the selected WebSphere MQ link sender channel.

WebSphere MQ link sender channel transmitter messages [Collection]

This pane displays the messages queued for transmission across the WebSphere MQ link sender channel.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel transmitters -> *WebSphere_MQ_sender_channel_name***

- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> Sender channel transmitter -> *WebSphere_MQ_sender_channel_name***
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel transmitters -> *WebSphere_MQ_link_sender_channel_name***
- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links -> *link_name* -> [Additional properties] Sender channel -> *sender_channel_link_name* -> Sender channel transmitter -> *WebSphere_MQ_link_sender_channel_name***

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Maximum messages displayed
The maximum number of messages retrieved and displayed.

Position
The position in the queue of the sender channel transmitter messages.

Identifier
The identifier of the message. Select this link to view the message detail panel for the selected message.

State The transaction state of the message (“Pending send”, “Pending acknowledgement”, “Committing”).

Transaction ID
The transaction identifier, if the message is locked under a transaction.

Target bus
The bus to which the message is targeted.

Target destination
The destination in the bus to which the message is targeted.

Approximate message length (bytes)
The approximate length of the message in bytes.

Buttons

Move	Move the selected available messages to the link exception destination. Only messages with a **State** of “Pending send” are moved.
Delete	Delete the selected available messages. Only messages with a **State** of “Pending send” are deleted.
Commit Pending Acknowledge Batch	Commit the pending acknowledgement message batch as being received successfully by the WebSphere MQ network. The messages are removed from the sender channel transmitter.

Rollback Pending Acknowledge Batch	Roll back the pending acknowledgement message batch as not being received successfully by the WebSphere MQ network. The messages are restored to the sender channel transmitter.
Refresh	Refresh the collection with the current set of messages on the sender channel transmitter. Only the number of messages up to the specified "Maximum displayed messages" are retrieved and displayed.

Service integration bus link routing properties [Settings]

The routing properties for a service integration bus link to a foreign service integration bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Additional Properties] Service integration bus link routing properties.

After a routing definition is configured, service integration bus links must be created between messaging engines in the services integration buses to be connected.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name by which the routing definition is known.

Required No
Data type String

UUID:

The universal unique identifier assigned by the system to the routing definition (virtual link) for administrative purposes.

Required No
Data type String

Inbound user ID:

The user ID applied to messages arriving from the foreign bus and used to authorize their access to destinations.

The inbound user ID is used to authorize individual messages arriving from the foreign bus to destinations in this bus. If this is not a secure bus, this property has no affect on messages. You might want to specify an inbound user ID:

- if the foreign bus is in a different security domain from this bus and user IDs from the foreign bus are not recognized in this bus

- to locally-control access of inbound messages to this bus.

If this is a secure bus and the foreign bus is not secure, and no inbound user ID is set, any inbound messages from the foreign bus will only be authorized to destinations that allow unauthenticated users access.

Dynamic updates to this property are effective immediately.

Required	No
Data type	String

Outbound user ID:

The user ID applied to messages sent to the foreign bus.

The outbound user ID replaces the user ID that identifies the source of a message in all messages being sent to the foreign bus. This user ID is also be used by the foreign bus to authorize the message to its destination if both buses are secure buses and the foreign bus has not overridden the user ID with its own inbound user ID.

Dynamic updates to this property are effective immediately.

Required	No
Data type	String

Additional Properties

Topic space mapping
The mapping between topic spaces in the local bus and topic spaces in the foreign bus.

Service integration bus links [Collection]

This pane displays links between this messaging engine and messaging engines in foreign service integration buses.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

An individual messaging engine can have links to messaging engines in a number of different foreign buses. However, you can have only one service integration bus link between a pair of buses.

Status can take the following values:

Table 218. Status values and definitions. The first column of the table lists the status values of the service integration bus link. The second column contains a brief definition of the status.

Status	Meaning
Starting	The service integration bus link is started on the local messaging engine but has no connection to the foreign bus. The service integration bus link is attempting to activate a connection to the foreign bus. The service integration bus link on the foreign bus must also be started to enable the successful activation of a connection between the buses.
Started	The service integration bus link is started on the local messaging engine and has an active connection to the foreign bus.
Stopped	The service integration bus link is stopped on the local messaging engine and there is no connection to the foreign bus.
Unknown	The administrative console cannot contact the server to determine the status.

Name The name of the service integration bus link. In order to work, the name must be the same as the name of the corresponding service integration bus link configured on the target foreign bus.

Description
An optional description for the service integration bus link, for administrative purposes.

UUID The universal unique identifier assigned by the system to the service integration bus link for administrative purposes.

Local messaging engine
The local messaging engine that this service integration bus link is hosted on.

Foreign messaging engine
The messaging engine on the foreign bus to which this service integration bus link connects.

Status
The runtime status of the service integration bus link.

Current outbound messages
The current total number of messages queued on the link transmitters to this service integration bus link.

Messages sent
The total number of messages sent on the link transmitters to this service integration bus link since the messaging engine was started.

Current inbound messages
The current total number of messages queued pending receipt on the link receivers for this service integration bus link.

Messages received
The total number of messages received on the link receivers for this service integration bus link since the messaging engine was started.

Buttons

Start	Start selected items.
Stop	Stop selected items.

Service integration bus links [Settings]

This pane displays links between this messaging engine and messaging engines in foreign service integration buses.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] Service integration bus links -> *link_name*

The messaging engine in the foreign bus must also have a service integration bus link to the local bus. For a connection to be active, the service integration bus links at both ends must be started.

- "Configuration tab"
- "Runtime tab" on page 2163

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the service integration bus link. In order to work, the name must be the same as the name of the corresponding service integration bus link configured on the target foreign bus.

Required	Yes
Data type	String

UUID:

The universal unique identifier assigned by the system to the service integration bus link for administrative purposes.

Required	No
Data type	String

Description:

An optional description for the service integration bus link, for administrative purposes.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Text area

UUID:

The universal unique identifier assigned by the system to the service integration bus link for administrative purposes.

Required	Yes
Data type	drop-down list

Local messaging engine:

The local messaging engine that this service integration bus link is hosted on.

Required Yes
Data type drop-down list

Foreign messaging engine:

The messaging engine on the foreign bus to which this service integration bus link connects.

Required Yes
Data type String

Initial state:

The initial state of the link, which shows whether the link is started automatically when the messaging engine is started.

Dynamic updates to this property are effective when the messaging engine is restarted or the service integration bus link is created. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Stopped
When the associated messaging engine is started, the gateway link is in a stopped state and cannot process any new requests for connections.

Started
When the associated messaging engine is started, the gateway link is in a started state and can process any new requests for connections.

Exception destination:

The destination for an inbound message when the service integration bus link cannot deliver the message to its target bus destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist.

Dynamic updates to this property are effective immediately.

Select a radio button as required to configure the exception destination that this service integration bus link uses:

- Select **None** to specify that the service integration bus link does not use an exception destination. Undeliverable messages are not rerouted to an exception destination and can block the processing of other messages waiting for delivery to the same destination. This option can be used to preserve message ordering.

- Select **System** to use the default exception destination. Messages that cannot be delivered to the bus destination are rerouted to the system default exception destination for the messaging engine that this link is assigned to: _SYSTEM.Exception.Destination*messaging_engine_name*.
- Select **Specify** and enter an exception destination to use the exception destination specified here. If the service integration bus link cannot use this exception destination, it uses the system exception destination.

Required	No
Data type	Radio button
Default	System

Prefer queue points local to this link's messaging engine:

When this check box is selected, the link prefers to send inbound messages to available queue points of target destinations that are located on the messaging engine on which the link is hosted.

When this check box is not selected, or if no local queue point is available for a target destination, the link workload balances the messages across all available queue points of the target destination. By default the check box is selected.

This option is supported on links running on WebSphere Application Server Version 7.0 or later. If you are running on an earlier version, the default behavior of preferring local queue points is applied.

Required	Yes
Data type	Boolean

Related Items

JAAS - J2C authentication data .
Specifies a list of user identities and passwords for Java 2 connector security to use. Refer to Java 2 Connector authentication data entry settings.

Foreign bus connection
The associated foreign bus connection for this service integration bus link.

Messaging engine
The remote messaging engine where this queue point is localized.

Link transmitters
For applications that use point-to-point messaging, there is one link transmission message point located on each messaging engine in the source bus. For applications that use publish/subscribe messaging, there is one link transmission message point located on each topic space in the source bus. The link transmitter acts as a transmission queue where produced messages are persisted before transmission across the inter-bus link to the foreign bus.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Status:

The runtime status of the service integration bus link.

Status can take the following values:

Table 219. Status values and definitions. The first column of the table lists the status values of the service integration bus link. The second column contains a brief definition of the status.

Status	Meaning
Starting	The service integration bus link is started on the local messaging engine but has no connection to the foreign bus. The service integration bus link is attempting to activate a connection to the foreign bus. The service integration bus link on the foreign bus must also be started to enable the successful activation of a connection between the buses.
Started	The service integration bus link is started on the local messaging engine and has an active connection to the foreign bus.
Stopped	The service integration bus link is stopped on the local messaging engine and there is no connection to the foreign bus.
Unknown	The administrative console cannot contact the server to determine the status.

Required No
Data type String

Transmitter/Receiver Queues

Link receivers

For applications that use point-to-point messaging, there is one link receiver for each messaging engine in the foreign bus. For applications that use publish/subscribe messaging, there is one link receiver for each topic space in the foreign bus. The link receiver acts as an inbound receiver queue for a message engine that has applications attached and that is producing messages across this service integration bus link.

Link transmitters

For applications that use point-to-point messaging, there is one link transmission message point located on each messaging engine in the source bus. For applications that use publish/subscribe messaging, there is one link transmission message point located on each topic space in the source bus. The link transmitter acts as a transmission queue where produced messages are persisted before transmission across the inter-bus link to the foreign bus.

Target inbound transport chain

The type of transport chain used for communication with the foreign bus.

The transport chain name must be the name of the transport chain as defined on the server on which the target messaging engine is hosted.

Dynamic updates to this property are effective when the link is restarted. Use the Runtime tab to check the current state.

Required No
Data type String

Bootstrap endpoints

A comma-separated list of endpoint triplets, with the syntax hostName:portNumber:chainName, used to connect to a bootstrap server. For example Merlin:7276:BootstrapBasicMessaging,Gandalf:5557:BootstrapSecureMessaging. If hostName is not specified, the default is localhost. If portNumber is not specified, the default is 7276. If chainName is not specified, the default is BootstrapBasicMessaging. Refer to the information center for more information.

This property is set in the same way as the **Provider endpoint** property in the JMS connection factory settings.

The port for the bootstrap endpoint is the port defined on the service integration bus endpoint address that is configured on the target application server on the foreign bus.

Dynamic updates to this property are effective when the link is restarted. Use the Runtime tab to check the current state.

You only have to modify this property if you have client applications running outside of an application server, or applications on a server in another cell, that want to use this connection factory to connect to the target service integration bus specified on the connection factory.

To use JMS destinations of the default messaging provider, an application connects to a messaging engine on the target service integration bus to which the destinations are assigned. For example, a JMS queue is assigned to a queue destination on a service integration bus.

Client applications running outside of an application server - for example, running in a client container or outside the WebSphere Application Server environment - cannot locate directly a suitable messaging engine to connect to in the target bus. Similarly, an application running on a server in one cell to connect to a target bus in another cell cannot locate directly a suitable messaging engine to connect to in the target bus.

In these scenarios, the clients (or servers in another bus) must complete a bootstrap process through a *bootstrap server* that is a member of the target bus. A bootstrap server is an application server running the SIB Service, but does not have to be running any messaging engines. The bootstrap server selects a messaging engine that is running in an application server that supports the required *target transport chain*. For the bootstrap process to be possible, you must configure one or more *provider end points* in the connection factory used by the client.

A bootstrap server uses a specific port and bootstrap transport chain. The port is the **SIB_ENDPOINT_ADDRESS** (or **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled), of the messaging engine that hosts the remote end of the link. Together with host name, these form the *endpoint address* of the bootstrap server.

The properties of a JMS connection factory used by an application control the selection of a suitable messaging engine and how the application connects to the selected messaging engine.

- If no security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7276 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used
- If security credentials are provided, then by default
  - If no host is specified then localhost is used
  - If no port is specified then port 7286 is used
  - If no bootstrap channel chain is specified then bootstrap transport chain called BootstrapBasicMessaging is used

Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

If you want an application to use a bootstrap server with a different endpoint address, you must specify the required endpoint address on the **Provider endpoints** property of the JMS connection factories that the client application uses. You can specify one or more endpoint addresses of bootstrap servers.

The endpoint addresses for bootstrap servers must be specified in every JMS connection factory that is used by applications outside of an application server. To avoid having to specify a long list of bootstrap servers, you can provide a few highly-available servers as dedicated bootstrap servers. Then you only have to specify a short list of bootstrap servers on each connection factory.

Note: When configuring a connection to a non-default bootstrap server, specify the required values for the endpoint address and use colons as separators.

For example: for a server assigned non-secure port 7278, on host boothost1, that uses the default transport chain BootstrapBasicMessaging:

```
boothost1:7278:BootstrapBasicMessaging
or
boothost1:7278
```

and for a server assigned secure port 7289, on host boothost2, that uses the predefined transport chain BootstrapTunneledSecureMessaging:

```
boothost2:7289:BootstrapTunneledSecureMessaging
```

The syntax for an endpoint address is as follows:

```
[ [host_name] [ ":" [port_number] [ ":" chain_name] ] ]
```

where:

host_name
: is the name of the host on which the server runs. It can be an IP address. For an IPv6 address, put square braces ([]) around *host_name* as shown in the example below:

  ```
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
  ```

  . If a value is not specified, the default is localhost.

  Note: For the IBM i platform, you must (at least) change the default host name from localhost to *your.server.name*.

port_number
: where specified, is one of the following addresses of the messaging engine that hosts the remote end of the link:

  - **SIB_ENDPOINT_ADDRESS** if security is not enabled
  - For secure connections, **SIB_ENDPOINT_SECURE_ADDRESS** if security is enabled.

  If *port_number* is not specified, the default is 7276.

  To find either of these values by using the administrative console, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Communications] Ports**.

chain_name
: is the name of a predefined bootstrap transport chain used to connect to the bootstrap server. If not specified, the default is BootstrapBasicMessaging.

  The following predefined bootstrap transport chains are provided:

  BootstrapBasicMessaging
  : This corresponds to the server transport chain InboundBasicMessaging (JFAP-TCP/IP)

  BootstrapSecureMessaging
  : This corresponds to the server transport chain InboundSecureMessaging (JFAP-SSL-TCP/IP)

  BootstrapTunneledMessaging
  : Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See **Servers -> Server Types -> WebSphere**

application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports.) This transport chain tunnels JFAP and uses HTTP wrappers.

BootstrapTunneledSecureMessaging
Before you can use this bootstrap transport chain, you must define a corresponding server transport chain on the bootstrap server. (See**Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] Messaging engine inbound transports**.) This transport chain tunnels JFAP and uses HTTP wrappers.

Specifying *host_name* : *chain_name* instead of *host_name* : : *chain_name* (with two colons) is incorrect. It is valid to enter nothing, or to enter any of the following: "a", "a:", ":7276", "::chain", and so on. The default value applies if you do not specify a value, but you must separate the fields with ":"s.

If you want to provide more than one bootstrap server, identify all the required endpoint addresses. Separate each endpoint address by a comma character. For example, to use the servers from the earlier example:

```
boothost1:7278:BootstrapBasicMessaging,
  boothost2:7289:BootstrapTunneledSecureMessaging,
  [2002:914:fc12:179:9:20:141:42]:7276:BootstrapBasicMessaging
```

Required	No
Data type	Text area

Authentication alias
The name of the authentication alias, used to authenticate access to the foreign bus.

You must have predefined a Java Platform, Enterprise Edition (Java EE) Connector Architecture (JCA) authentication alias.

Dynamic updates to this property are effective when the link is restarted. Use the Runtime tab to check the current state.

Modified aliases are only visible after a server restart.

Required	No
Data type	drop-down list

Subscriptions [Collection]
The active subscriptions for the topic space.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name* -> [Message points] Publication points -> *publication_point_name* -> Runtime -> Subscriptions.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name by which the active subscription is known for administrative purposes.

Identifier
The identifier by which this destination is known for administrative purposes.

Topic The name of the topic that this subscription is for.

Queue Depth
The number of messages currently associated with this subscription.

Buttons

Delete	Delete the selected items.

Broker profile subscriptions [Collection]

The list of current broker subscriptions for this broker profile.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name* -> Runtime > **Subscriptions**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Topic name
The topic name for this subscription.

Topic space
The topic space for this subscription.

Direction
Whether the subscription is for message flows to or from WebSphere MQ.

Broker stream queue
The name of the WebSphere MQ broker stream queue that the topic publishes messages to.

Subscription point
The name of the WebSphere MQ broker subscription point that the topic consumes messages from.

Message count
The count of messages for this subscription.

Status
The current status of this subscription.

Buttons

Unsubscribe	Cancel all subscriptions.

Subscriptions [Settings]

The active subscriptions for the topic space.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name* -> [Message points] Publication points -> *publication_point_name* -> Runtime -> Subscriptions -> *subscription_name*.

- "Runtime tab"

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Buttons

Refresh	Refresh the number of messages.

General Properties

Name:

The name by which the active subscription is known for administrative purposes.

Required No
Data type String

Identifier:

The identifier by which this destination is known for administrative purposes.

Required No
Data type String

Topic:

The name of the topic that this subscription is for.

Required No
Data type Text area

Selector:

The text string that must be present in a message for the mediation to process the message.

Required No
Data type String

Current message depth:

The number of messages on the message point.

Required	No
Data type	String

Subscriber ID:

Required	No
Data type	String

Additional Properties

Messages
Messages being handled by this subscription.

Known remote subscription points
The remote messaging engines that have remote consumers connected to this subscription point.

Temporary destination prefixes [Collection]

A temporary destination prefix is a user-defined string that is used to create a temporary destination. When messaging security is enabled, users and groups require authority to create messages and send them to temporary destinations. The authority is configured in the temporary destination prefix.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage temporary destination prefix access roles

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

A list of all the temporary destination prefixes defined for the selected service integration bus is displayed. By default, the list is empty. Use this pane to create a new temporary destination prefix, and define its role type assignments, or remove existing temporary destination prefixes. You can also manage the role type assignments for existing temporary destination prefixes:

- To act on a single temporary destination prefix, click its name in the list.
- To act on more than one temporary destination prefix, select the check box next to the name of each destination you want to act on, then click **Manage Access Roles**.

Temporary destination prefix
The name of each temporary destination prefix defined for the selected bus.

Buttons

Manage Access Roles	Click to view and manage users and groups assigned to the access role types for the selected resources.

Add	Click to create a temporary destination prefix, and add users and groups to it.
Remove	Click to remove selected users and groups from all the role types for this resource.

Temporary destination prefixes [Settings]

This pane displays the role type assignments for the selected temporary destination prefixes.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage temporary destination prefix access roles -> *temporary_destination_prefix* > **Manage access roles**

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

A list of users and groups that have been added to access roles for the selected temporary destination prefixes is displayed. The access role information for the selected temporary destination prefix is contained within an expandable section. The access roles for temporary destination prefixes are creator and receiver. Users and groups in the creator role can create a new temporary destination prefix. You cannot remove users and groups from the creator role.

Use this pane to complete the following tasks:

- View users and groups that have creator and sender roles for a selected temporary destination prefix.
- Add or remove users and groups in the sender role for a selected temporary destination prefix.

General Properties

Select A check box that you can use to select the users and groups for which you want to manage access roles.

Name The name of the user or group that has an access role for the selected resource. If the user is a group member, the user ID and the group name is displayed.

Type The type of the user or group. There are three types of user or group: “user”, “group” and “member”. A user that inherits its access roles from a group has the type “member”.

Sender
Whether a user, group or member is in the sender role for a selected resource.

Creator
Read only. The user or group is in the creator role for a selected temporary destination prefix. You cannot change this role type assignment.

Security access roles

In the administrative console, access role icons are used to represent whether a user or a group is in a particular access role. You can click an icon to add or remove selected users and groups to a particular access role for a selected resource.

An access role icon has three states:

- Access role type set.
- Access role type not set.
- Access role type inherited from group.

The following table describes how the access role icons represent these states, and how to change between them:

Table 220. Interacting with access role icons

Access role icon	Access role assignment state	User action
	Role type not set.	Click to change to role type set  .
	Role type set.	Click to change to role type not set. The icon changes to role type not set  if the user or group does not inherit access roles, or to role type inherited  if the role type does inherit access roles.
	Role type inherited from group.	Click to change to role type set  .
	Role type not set for a group. The group to which a user belongs does not have a role type.	Read only.
	Role type set for a group. The group to which a user belongs has a role type.	Read only.
	Role type not applicable.	Read only.

Topic Mapping [Collection]

The mapping between a topic on the service integration bus and a stream queue and subscription point provided by a WebSphere MQ broker.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name* -> [Additional Properties] Topic mappings.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Dynamic updates to this list are effective immediately.

Topic name
The name of the topic on the service integration bus. The name must be the same as the topic name on the message broker in a WebSphere MQ network.

Topic space
The name of the topic space that contains the topic.

Direction
Whether the mapping is for publishing message flows in both directions, or only to, or only from, WebSphere MQ.

Broker stream queue
The name of the stream queue at the message broker in a WebSphere MQ network.

Subscription point
The name of the WebSphere MQ broker subscription point that the topic consumes messages from.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Topic Mapping [Settings]

The mapping between a topic on the service integration bus and a stream queue and subscription point provided by a WebSphere MQ broker.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name* -> [Additional Properties] Topic mappings -> *mapping_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Topic name:

The name of the topic on the service integration bus. The name must be the same as the topic name on the message broker in a WebSphere MQ network.

You can specify wildcard characters in topic name strings.

Required Yes
Data type String

Topic space:

The name of the topic space that contains the topic.

Required No
Data type drop-down list

Direction:

Whether the mapping is for publishing message flows in both directions, or only to, or only from, WebSphere MQ.

Required No

Data type drop-down list

Range

Bi-directional
Messages flow in both directions between the bus and WebSphere MQ.

To WebSphere MQ
Messages flow only from the bus to WebSphere MQ.

That is, from WebSphere Application Server to a message broker in the WebSphere MQ network.

From WebSphere MQ
Messages flow only to the bus from WebSphere MQ.

That is, from a message broker in the WebSphere MQ network to WebSphere Application Server.

Broker stream queue:

The name of the stream queue at the message broker in a WebSphere MQ network.

The broker stream queue in this instance is a queue on the WebSphere MQ queue manager to which the message broker is connected. This queue is being used as the input node for a message flow containing a publication node. Messages sent to this queue are processed by the message broker, then published to applications that have subscribed on the topic specified in the message.

Stream names are case sensitive.

After you type a new name, then save your changes, the name becomes available for selection in the drop-down list.

Required No
Data type Custom

Subscription point:

The name of the WebSphere MQ broker subscription point that the topic consumes messages from.

Type the name of the WebSphere MQ message broker subscription point from which the service integration bus receives messages.

After you type a new name, then save your changes, the name becomes available for selection in the drop-down list.

Required No
Data type Custom

Topic [Settings]

This pane displays the role type assignments for a topic. You can use this pane to add new assignments, and to modify and remove existing assignments.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage topic access roles -> *topic_space_name* > *topic_name*.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

This pane lists users and groups that have been assigned role types on the selected topic. The permitted access roles for topics are sender and receiver. Use this panel to view existing topic role assignments, add new users and groups to topic access roles, and remove users and groups from topic access roles.

The information for the selected topic is contained within a section that you can expand to display all the users and groups that have been assigned role types for the topic. The following options are available:

Inherit sender role from parent topic
Select the check box if you want the topic to inherit sender role assignments from the parent topic.

Inherit receiver role from parent topic
Select the check box if you want the topic to inherit receiver role assignments from the parent topic.

General Properties

Select A check box that you can use to select the users and groups for which you want to manage access roles.

Name The name of the user or group that has an access role for the selected resource. If the user is a group member, the user ID and the group name is displayed.

Type The type of the user or group. There are three types of user or group: “user”, “group” and “member”. A user that inherits its access roles from a group has the type “member”.

Sender
Whether a user, group or member is in the sender role for a selected resource.

Receiver
Whether a user, group or member is in the receiver role for a selected resource.

Security access roles

In the administrative console, access role icons are used to represent whether a user or a group is in a particular access role. You can click an icon to add or remove selected users and groups to a particular access role for a selected resource.

An access role icon has three states:

- Access role type set.
- Access role type not set.
- Access role type inherited from group.

The following table describes how the access role icons represent these states, and how to change between them:

Table 221. Interacting with access role icons

Access role icon	Access role assignment state	User action
	Role type not set.	Click to change to role type set  .
	Role type set.	Click to change to role type not set. The icon changes to role type not set  if the user or group does not inherit access roles, or to role type inherited  if the role type does inherit access roles.
	Role type inherited from group.	Click to change to role type set  .
	Role type not set for a group. The group to which a user belongs does not have a role type.	Read only.
	Role type set for a group. The group to which a user belongs has a role type.	Read only.
	Role type not applicable.	Read only.

Buttons

Add	Click to add users and groups to this resource.
Remove	Click to remove selected users and groups from all the role types for this resource.

Topic space map entries [Collection]

The mapping between a topic space in the local bus and a topic space in the foreign bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Additional Properties] Service integration bus link routing properties -> [Additional Properties] Topic space map entries.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. For publications to flow from the local topic space into the foreign bus, an equivalent topic space mapping is required by the foreign bus.

Dynamic updates to this list are effective immediately.

Local topic space
The name of the topic space on this (local) bus that is mapped to the remote topic space on the foreign bus.

Remote topic space
The name of the topic space on the foreign bus that is mapped to the local topic space.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Topic space map entries [Settings]

The mapping between a topic space in the local bus and a topic space in the foreign bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Additional Properties] Service integration bus link routing properties -> [Additional Properties] Topic space map entries -> *map_entry_name*.

A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. For publications to flow from the local topic space into the foreign bus an equivalent topic space mapping is required by the foreign bus.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Local topic space:

The name of the topic space on this (local) bus that is mapped to the remote topic space on the foreign bus.

Required Yes
Data type Custom

Remote topic space:

The name of the topic space on the foreign bus that is mapped to the local topic space.

Required Yes
Data type String

Topic space mapping [Settings]

The mapping between topic spaces in the local bus and topic spaces in the foreign bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Publish/subscribe broker profiles -> *profile_name* -> [Additional Properties] Topic space mapping.

A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. For publications to flow from the local topic space into the foreign bus an equivalent topic space mapping is required by the foreign bus.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Description:

An optional description for the topic space mapping, for administrative purposes.

Dynamic updates to this property are effective immediately.

Required	No
Data type	Text area

Additional Properties

Topic space map entries
The mapping between topic spaces in the local bus and topic spaces in the foreign bus.

Topic space [Settings]

A topic space is a location for publish/subscribe messaging.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *topic_space_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The identifier by which this destination is known for administrative purposes.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this destination for administrative purposes.

Required	No
Data type	String

Type:

Whether this bus destination is for a queue, topic space, or some other type of destination.

A topic space for publish/subscribe messaging.

Required	No
Data type	String

Description:

An optional description for the bus destination, for administrative purposes.

Required	No
Data type	Text area

Mediation:

The name of the mediation that mediates this destination.

Required	No
Data type	String

Enable producers to override default reliability:

Select this option to enable producers to override the default reliability that is set on the destination.

Required	No
Data type	Boolean

Default reliability:

The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

Required	No
Data type	drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Maximum reliability:

The maximum reliability of messages accepted by this destination.

Required No
Data type drop-down list
Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Default priority:

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

Required	No
Data type	Integer
Range	0 through 9

Exception destination:

Use these properties to define what happens to any messages that cannot be delivered to this destination.

None:

The bus destination does not use an exception destination and undeliverable messages are not rerouted to an exception destination.

Attempts to redeliver the message continue, up to the maximum failed deliveries limit set for the bus destination. Then, attempts to redeliver the message continue with a time interval between retry attempts. This interval is either the Default blocked destination retry interval of the messaging engine that is associated with this destination, or the Blocked retry timeout that is set for this destination. The Default blocked destination retry interval value can be used by all queue and topic destinations associated with this messaging engine. To set a time interval specifically for this destination, select **Override messaging engine blocked retry timeout default**, then enter a blocked retry timeout value for this destination.

Required	No
Data type	Radio button
Default	Not selected

Override messaging engine blocked retry timeout default:

Override the blocked queue retry interval configured on the messaging engine owning the destination.

Select this property to set the blocked retry timeout for this destination. This property is available only when **None** is selected for the exception destination.

Required	Yes, if **None** is selected.
Data type	Boolean
Default	Unchecked. The Default blocked destination retry interval value of the associated messaging engine is used.

Blocked retry timeout in milliseconds:

When no exception destination is configured, the time interval to apply between retry attempts, after the maximum failed deliveries limit is reached, for this destination.

This property is available only when **Override messaging engine blocked retry timeout default** is selected in the exception destination properties.

Required	Yes, if Override messaging engine blocked retry timeout default is checked.
Data type	Integer

System:

The bus destination uses the system default exception destination.

Undeliverable messages are routed to the system default exception destination of the messaging engine that detects the problem: _SYSTEM.Exception.Destination.*messaging_engine_name*.

Required	No
Data type	Radio button
Default	Selected

Specify:

Select this property to configure a specific exception destination.

The exception destination must be a queue, on the same bus or a foreign bus, and must exist when the exception destination processing is configured.

Required	No
Data type	Radio button
Default	Not selected

Maximum failed deliveries per message:

The maximum number of failed attempts to process a message. After this number of failed attempts, if an exception destination is configured, the message is forwarded from the intended destination to its exception destination. If an exception destination is not configured, a time interval between retry attempts is applied.

This interval is either the Default blocked destination retry interval of the messaging engine that is associated with this destination, or the Blocked retry timeout that is set for this destination.

Required	Yes, if an exception destination has been configured.
Data type	Integer
Default	5
Range	0 through 2147483647

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

Required	No
Data type	Boolean

Receive allowed:

Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination.

Required	No
Data type	Boolean

Maintain strict message order:

Enabling this option will maintain the strict ordering of messages for this destination.

Required	No
Data type	Custom

Reply destination:

The name of a destination to be appended to any non-empty reverse routing path of messages sent to this destination.

Required	No
Data type	String

Reply destination bus:

The bus on which the reply destination exists.

Required	No
Data type	String

Allow auditing of Topic access checks:

This option is only available if bus and cell level auditing are enabled. Check to enable auditing of Topic access checks.

Required	No
Data type	Boolean

Topic access check required

Whether or not authorization checks are required for access to topics.

When security is on, authorization checks are always performed at the topic space level. To add additional control, you can select this property to enable authorization checks at the topic level.

Required	No
Data type	Boolean

Message points

Publication points

A publication point is created on each messaging engine in the bus when a publish/subscribe destination is created. The publication point on a messaging engine is used to hold messages published by applications connected to that messaging engine until they are delivered to subscribers.

Mediation points

The mediation points for the topic space. The locations in the messaging engine at which messages on the topic space are mediated.

Additional Properties

Context properties

Context information passed to the mediation.

Related Items

Application resources topology
A expandable tree view of all applications and messaging resources that reference the current destination.

Audit Service
Configure the global audit settings

Topic spaces [Collection]

Topics are defined hierarchically within topic spaces. To view the access roles for a topic, first select the topic space that contains the topic to be viewed.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage topic access roles.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

The Topic spaces pane displays a list of topic spaces defined on the selected bus. A topic space is a hierarchy of topics used for publish/subscribe messaging. Use this pane to select the topic space that contain the topics for which you want to add, remove or modify role type assignments.

Topic space
The name of each topic space defined for the selected bus.

Topics [Collection]

Topics are defined hierarchically within topic spaces. To view a topic's access roles, select the topic from the hierarchical list of topics below.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage topic access roles

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.The Topics pane displays a list of topics defined for the selected topic space. The topics are displayed hierarchically beneath the root (/) topic. You can expand the topic space name to display all the topics, or collapse it to display the root topic only. By default, a topic space does not contain any topics. Use **Add** to add a new topic to the selected topic space, and to define its role type assignments.

Topic The name of each topic defined in the selected topic space.

Buttons

Add	Click to create a topic, and add users and groups to it.
Show all	Click to expand a section to display a hierarchical list of all the topics defined in the topic space, starting with the root (/) topic.
Hide all	Click to collapse a section to display the root (/) topic only.

Topics [Collection]

The topics that have been subscribed to from this remote messaging engine.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Remote message points] Remote publication points** > > ***identifier_name*** **-> [Outbound properties] Topics**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Topic View the outbound topics for this publication point.

Buttons

Clear all	Clear all messages from this message point.

Unknown user or group [Settings]

This panel lists users and groups that do not exist in the user repository but have access role definitions.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage users and groups not known to the user repository -> *user_name* > **Remove roles**

The Unknown users and groups pane lists access roles for bus resources that have been assigned to selected unknown users and groups.

The access role information for each unknown user or group is contained within an expandable section.

General Properties

Resource
The selected unknown user or group has access roles for this bus resource.

Sender
Whether a user, group or member is in the sender role for a selected resource.

Receiver
Whether a user, group or member is in the receiver role for a selected resource.

Browser
Whether a user, group or member is in the browser role for a selected resource.

Creator
Whether a user, group or member is in the creator role for a selected resource.

Bus connector
Whether a user, group or member is in the bus connector role for a selected resource.

Security access roles

In the administrative console, access role icons are used to represent whether a user or a group is in a particular access role. You can click an icon to add or remove selected users and groups to a particular access role for a selected resource.

An access role icon has three states:

- Access role type set.
- Access role type not set.
- Access role type inherited from group.

The following table describes how the access role icons represent these states, and how to change between them:

Table 222. Interacting with access role icons

Access role icon	Access role assignment state	User action
	Role type not set.	Click to change to role type set  .
	Role type set.	Click to change to role type not set. The icon changes to role type not set  if the user or group does not inherit access roles, or to role type inherited  if the role type does inherit access roles.
	Role type inherited from group.	Click to change to role type set  .
	Role type not set for a group. The group to which a user belongs does not have a role type.	Read only.
	Role type set for a group. The group to which a user belongs has a role type.	Read only.
	Role type not applicable.	Read only.

Buttons

Remove	Click to remove access roles from the selected user or group.

Unknown users and groups [Collection]

The users and groups displayed here are not defined in the user registry, but do have roles defined. This could be because they were removed from the user repository after the role assignments were made.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Manage users and groups not known to the user repository

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

This pane displays a list of user and group names that were previously created in the user repository, and granted access roles to selected resources, but the user and group names have since been removed from the user repository. Use this pane to remove the role type assignments from one or more users and groups that no longer exist in the user repository:

- To work with a single unknown user or group name, click the user or group name.
- To work with more than one unknown user or group name, select the check box in the **Select** column for each unknown user or group name, and click **Remove roles**.

Name The name of a user that has an access role definition, but is not defined in the user registry.

Type Whether the user is a user, a group, or a group member.

Buttons

Remove roles	Click to remove access roles from the selected user or group.

Users and groups in the bus connector role [Collection]

Users in the bus connector role are able to connect to the bus to perform messaging operations. Users can have this role either by specifically having that role, or because they are in a group with that role.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Users and groups in the bus connector role.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

Name The name of the user or group in the bus connector role.

Type The type of the user or group in the bus connector role.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Add a user or group to the bus connector role [Settings]

Create a user or group in the bus connector role.

To view this page in the console, click the following path:

Service integration -> Buses -> *security_value* -> [Authorization Policy] Users and groups in the bus connector role -> *member_name*.

In the path, *security_value* is either **Enabled** if messaging security is enabled, or **Disabled** if messaging security is not enabled.

When messaging security is enabled, users must be authorized to connect to a local service integration bus before they can carry out messaging operations. To authorize a user or a group to connect to a local bus, you add the user or group name to the bus connector role for the bus. After the user connects to the bus, then can access local bus destinations, and send messages to destinations on foreign buses.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Bus Connector Role:

The bus connector role. Users in this role are able to connect to the bus to perform messaging operations.

Select one of the options provided to choose the bus connector role. A user in the bus connector role has the authority to connect to the local service integration bus, and can carry out messaging operations at destinations on the bus.

Required	No
Data type	Custom

Range

Group name
Give a specific group the bus connector role. All users in the group can connect to the bus, and use it to undertake messaging operations. Type the name of the group in the field provided.

User name
Give a user the bus connector role. Users with the specified user ID can connect to the bus, and use it to perform messaging operations. Type the user ID in the field provided.

Server - Allow servers to connect to the bus
Give all application servers the bus connector role. This represents the identity of a WebSphere Application Server. This can be used by message-driven beans that want message delivery, without specifying an authentication alias, in a secured bus.

All Authenticated - Allow all authenticated users to connect to the bus
Give only *authenticated* users the bus connector role. All users that have authenticated to the bus can use it to undertake messaging operations. This results in the group “AllAuthenticated” being added to the authorization model.

Everyone - Allow unauthenticated users to connect to the bus
Give all users the bus connector role. All users that have connected to the bus can use it to undertake messaging operations. If this option is selected, then users are able to connect to the bus without authenticating. These users are treated as anonymous users.

Web service [Settings]

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Destination resources] Destinations -> *web_service_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The identifier by which this destination is known for administrative purposes.

Required No
Data type String

UUID:

The universal unique identifier assigned by the system to this destination for administrative purposes.

Required	No
Data type	String

Type:

Whether this bus destination is for a queue, topic space, or some other type of destination.

Required	No
Data type	String

Description:

An optional description for the bus destination, for administrative purposes.

Required	No
Data type	Text area

Mediation:

The name of the mediation that mediates this destination.

Required	No
Data type	String

Default reliability:

The reliability assigned to a message produced to this destination when an explicit reliability has not been set by the producer.

Required	No
Data type	drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Maximum reliability:

The maximum reliability of messages accepted by this destination.

Required No
Data type drop-down list
Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Enable producers to override default reliability:

Select this option to enable producers to override the default reliability that is set on the destination.

Required No
Data type Boolean

Default priority:

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

Required No
Data type Custom

Exception destination:

Use these properties to define what happens to any messages that cannot be delivered to this destination.

Use this property to define what happens to any messages that cannot be delivered to this destination.

By default, such messages are routed to the system default exception destination of the messaging engine that discovers the problem: _SYSTEM.Exception.Destination.*engine_name*.

If you want messages to be sent to another exception destination, select Specify then type the exception destination name. The exception destination must be a queue, on the same bus or a foreign bus, and must exist when the destination is created.

If you do not want undeliverable messages to be sent to an exception destination, select None.

Required No
Data type String and Boolean

Send allowed:

Clear this option (setting it to false) to stop producers from being able to send messages to this destination.

Required No
Data type Boolean

Receive allowed:

Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination.

Required No
Data type Boolean

Receive exclusive:

Select this option to allow only one consumer to attach to each message point. If this option is not selected multiple consumers will be allowed to attach and receive messages from each message point.

Required No
Data type Boolean

Maintain strict message order:

Enabling this option will maintain the strict ordering of messages for this destination.

Required	No
Data type	Custom

Additional Properties

Context properties
Context information passed to the mediation.

Message points

Mediation points
A mediation point is a location in a messaging engine at which messages are mediated. A mediation point is created when a mediation is associated with a bus destination.

Client connections [Collection]

The connection between a WebSphere MQ client and the bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links -> *link_name* [Additional Properties] Client connections.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This panel lists all the connections for clients that use the related WebSphere MQ client link. You can select one or more connections to display more detail about their runtime status, or to stop them while leaving other connections active.

IP address
The TCP/IP IP address of the WebSphere MQ client.

Status
The runtime status of the WebSphere MQ client connection.

WebSphere MQ client connection [Settings]

The connection between a WebSphere MQ client and the bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links -> *link_name* [Additional Properties] Client connections -> *connection_name*.

• “Runtime tab” on page 2194

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

IP address:

The TCP/IP IP address of the WebSphere MQ client.

Required No
Data type String

Status:

The runtime status of the WebSphere MQ client connection.

Required No
Data type drop-down list
Range

Inactive
The WebSphere MQ client link is ready and enabled to receive inbound connections from clients, but no client connections have yet been established.

Stopped
The WebSphere MQ client link is in a stopped state and cannot process any new requests for inbound connections from clients.

Starting
The WebSphere MQ client link has received an inbound connection from a client, and is in the process of starting this connection.

Binding
The WebSphere MQ client link is performing channel negotiation and is not yet ready to transfer messages.

Running
The WebSphere MQ client link has an active session with a client, and data can be flowing between the client and the WebSphere MQ client link.

Stopping
The WebSphere MQ client link, or a connection with that link, is in the process of being stopped.

Multiple
The WebSphere MQ client link has established sessions with several clients, and the sessions can be in different states: Starting, Running, Stopping, or Stopped.

Number of messages sent:

The number of messages sent on the WebSphere MQ client link over the connection to a specific client.

Required	No
Data type	String

Number of messages received:

The number of messages received on the WebSphere MQ client link over the connection from a specific client.

Required	No
Data type	String

Number of buffers sent:

The number of message buffers sent on the WebSphere MQ client link over the connection to a specific client.

Required	No
Data type	String

Number of buffers received:

The number of message buffers received on the WebSphere MQ client link over the connection from a specific client.

Required	No
Data type	String

Number of bytes sent:

The number of bytes sent on the WebSphere MQ client link over the connection to a specific client.

Required	No
Data type	String

Number of bytes received:

The number of bytes received on the WebSphere MQ client link over the connection from a specific client.

Required	No
Data type	String

Channel start time:

The time at which the client connection channel was started.

Required	No
Data type	String

Channel start date:

The date on which the client connection channel was started.

Required	No
Data type	String

Last message send time:

The time at which the last message was sent on the client connection channel.

Required	No
Data type	String

Last message send date:

The date on which the last message was sent on the client connection channel.

Required	No
Data type	String

Last message receive time:

The time at which the last message was received on the client connection channel.

Required	No
Data type	String

Last message receive date:

The date on which the last message was sent on the client connection channel.

Required	No
Data type	String

Heartbeat interval:

The time, in seconds, between heartbeat flows passed from the WebSphere MQ client link to the WebSphere MQ classes for JMS that process JMS requests issued by the client application. This allows the WebSphere MQ client link to handle situations where the client connection created by the WebSphere MQ classes for JMS fails during a request.

Required	No
Data type	String

Maximum message length:

The maximum message length, in bytes, that can be transmitted using the WebSphere MQ client link. This is compared with the value for the partner WebSphere MQ client channel and the actual maximum used is the lower of the two values.

Required	No
Data type	String

Stop requested:

Whether or not a manual stop request has been made for the WebSphere MQ client link connection.

Required No
Data type drop-down list
Range

true The channel is in doubt about which messages have been committed by WebSphere MQ for the unit of work that it has sent.

false The channel is not in doubt about which messages have been committed by WebSphere MQ.

Local address:

The local address of the WebSphere MQ client connection.

Required No
Data type String

Number of API calls serviced:

The number of API calls serviced for the WebSphere MQ client connection.

Required No
Data type String

WebSphere MQ client link advanced properties [Settings]

Advanced configurable properties, such as message reliability and broker queue names, for the WebSphere MQ client link.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links -> *link_name* -> [Advanced properties] > ***property_name***.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Inbound persistent message reliability:

The acceptable reliability of message delivery for inbound persistent message flows from WebSphere MQ through this WebSphere MQ client link, from Reliable to Assured, in order of increasing reliability.

Required No
Data type drop-down list
Range

Reliable
Messages might be discarded when a messaging engine fails.

Assured
Messages are not discarded.

Inbound nonpersistent message reliability:

The acceptable reliability of message delivery for inbound nonpersistent message flows from WebSphere MQ through this WebSphere MQ link, from Reliable to Assured, in order of increasing reliability.

Required No
Data type drop-down list
Range

Best effort
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable
Messages are discarded when a messaging engine stops or fails.

Broker control queue:

The name of the message broker control queue to which all command messages (except publications and requests to delete publications) are sent.

Publisher and subscriber applications, and other brokers, send all command messages (except publications and requests to delete publications) to this queue.

Required No
Data type Text

Broker publication queue:

The name of the message broker publication queue to which all publication messages for the default stream are sent.

This is the name of the broker input queue (stream queue) that receives all publication messages for the default stream. Applications can also send requests to delete publications on the default stream to this queue.

Required	No
Data type	Text

Broker subscription queue:

The name of the message broker subscription queue from which nondurable subscription messages are retrieved.

This is the broker queue from which nondurable subscription messages are retrieved. The subscriber specifies the name of the queue when it registers a subscription.

Required	No
Data type	Text

Broker durable subscription queue:

The name of the message broker durable subscription queue from which durable subscription messages are retrieved.

This is the broker queue from which durable subscription messages are retrieved. The subscriber specifies the name of the queue when it registers a durable subscription.

Required	No
Data type	Text

Broker connection consumer subscription queue:

The name of the message broker connection consumer subscription queue from which nondurable subscription messages are retrieved for a connection consumer request.

This is the name of the broker queue from which nondurable subscription messages are retrieved for a ConnectionConsumer request.

Required	No
Data type	Text

Broker connection consumer durable subscription queue:

The name of the message broker connection consumer durable subscription queue from which nondurable subscription messages are retrieved for a connection consumer request.

This is the name of the broker queue from which durable subscription messages are retrieved for a ConnectionConsumer request.

Required	No
Data type	Text

Default topic space:

The name of the default topic space for the WebSphere MQ client link.

This topic space is used for publish/subscribe messages sent to Version 5.1 JMS topics for which the JMS topic connection factory connects to the node on which this WebSphere MQ client link is configured.

Required	No
Data type	Text

WebSphere MQ client links [Collection]

A WebSphere MQ client link presents the messaging engine, and therefore the bus, as a WebSphere MQ queue manager to which WebSphere MQ clients can attach. This behavior enables WebSphere Application Server Version 5 JMS clients to use messaging resources on the bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

If WebSphere MQ functionality has been disabled at any scope, an informational message indicating that WebSphere MQ has been disabled is displayed. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The name of the WebSphere MQ client link.

Description
An optional description for the WebSphere MQ client link, for administrative purposes.

MQ channel name
The name of the channel for the WebSphere MQ client link, used to flow messages between WebSphere MQ clients and the bus.

Queue manager name
The name of the WebSphere MQ queue manager on which the WebSphere MQ sender channel, that is connected to this MQ link receiver channel connection instance, is running.

Default queue manager
Whether or not this is the default queue manager for the WebSphere MQ clients.

Status
The runtime status of the WebSphere MQ client link.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.
Start	Start selected items. This button does not work if the selected WebSphere MQ client link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Table 223. Stop modes. The table contains information about the target states and the corresponding stop mode behavior. There are two target states such as inactive and stopped, and there are two stop modes such as quiesce and force. The two rows in the table represent the two target states, and the two columns describes the two stop mode behaviors for each of the target state.

		Stop mode	
		Quiesce	Force
Target state	Inactive	The sender channel becomes inactive either when it has finished processing its current batch, or when it reaches a heartbeat interval.	The sender channel immediately becomes inactive.
	Stopped	The sender channel becomes stopped either when it has finished processing its current batch, or when it reaches a heartbeat interval.	The sender channel immediately becomes stopped.

This button does not work if the selected WebSphere MQ client link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

WebSphere MQ client link [Settings]

A WebSphere MQ client link presents the messaging engine, and therefore the bus, as a WebSphere MQ queue manager to which WebSphere MQ clients can attach. This behavior enables WebSphere Application Server Version 5 JMS clients to use messaging resources on the bus.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional Properties] WebSphere MQ client links -> *link_name*.

If WebSphere MQ functionality has been disabled at any scope, an informational message indicating that WebSphere MQ has been disabled is displayed. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

- “Runtime tab”
- “Configuration tab” on page 2202

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Status:

The runtime status of the WebSphere MQ client link.

Required No

Data type drop-down list

Range

Inactive
The WebSphere MQ client link is ready and enabled to receive inbound connections from clients, but no client connections have yet been established.

Stopped
The WebSphere MQ client link is in a stopped state and cannot process any new requests for inbound connections from clients.

Starting
The WebSphere MQ client link has received an inbound connection from a client, and is in the process of starting this connection.

Binding
The WebSphere MQ client link is performing channel negotiation and is not yet ready to transfer messages.

Running
The WebSphere MQ client link has an active session with a client, and data can be flowing between the client and the WebSphere MQ client link.

Stopping
The WebSphere MQ client link, or a connection with that link, is in the process of being stopped.

Multiple
The WebSphere MQ client link has established sessions with several clients, and the sessions can be in different states: Starting, Running, Stopping, or Stopped.

Additional Properties

Client connections
The client connections that exist on the WebSphere MQ client link.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the WebSphere MQ client link.

Required Yes
Data type String

UUID:

The universal unique identifier assigned by the system to this WebSphere MQ client link for administrative purposes.

Required	No
Data type	String

Description:

An optional description for the WebSphere MQ client link, for administrative purposes.

Required	No
Data type	Text area

MQ channel name:

The name of the channel for the WebSphere MQ client link, used to flow messages between WebSphere MQ clients and the bus.

The name of the WebSphere MQ server connection channel that this client link will use to represent itself to clients. It must exactly match the value specified by the client for a connection to be established. For Version 5 embedded messaging clients a value of "WAS.JMS.SVRCONN" must be specified. You should not define two MQ client links that share both channel name and queue manager name. However , it is possible to have client links with identical channel names if they specify different queue manager names.

.

Required	Yes
Data type	String

Queue manager name:

The name of the WebSphere MQ queue manager on which the WebSphere MQ sender channel, that is connected to this MQ link receiver channel connection instance, is running.

The WebSphere MQ client link for use by JMS applications on WebSphere Application Server Version 5.1 has the queue manager name `WAS_nodeName_jmsserver`. You should not use this name on any other WebSphere MQ client link that is assigned to a messaging engine on the same node.

Required	Yes
Data type	String

Default queue manager:

Whether or not this is the default queue manager for the WebSphere MQ clients.

If a client does not specify the name of the queue manager it is to connect to, it will be connected to an MQ client link with a matching channel name that has been marked as being the default, if one matching this criterion can be found.

Required	No
Data type	Boolean

Maximum message size:

The maximum message length, in bytes, that can be transmitted using the WebSphere MQ client link.

Required	No
Data type	Long
Range	0 through 104857600

Heartbeat interval:

The time, in seconds, to wait before checking that a client requesting an operation is still active.

This heartbeat interval allows a sending channel to verify that the receiving channel is still active just before committing a batch of messages. If the receiving channel is not active, the batch can be backed out rather than becoming indoubt, as would otherwise be the case. By backing out the batch, the messages remain available for processing so they can, for example, be redirected to another channel.

If the sending channel has had a communication from the receiving channel within the batch heartbeat interval, the receiving channel is assumed to be still active, otherwise a “heartbeat” signal is sent to the receiving channel to check.

Required	No
Data type	Integer
Range	0 through 999999 A value of 0 (zero) indicates that the heartbeat mechanism is not used.

Initial state:

Whether the WebSphere MQ client link is started or stopped when the associated messaging engine is first started. Until started, the WebSphere MQ client link is unavailable.

Required	No
Data type	drop-down list
Range	**Stopped** When the associated messaging engine is started, the WebSphere MQ client link is in a stopped state and cannot process any new requests for inbound connections from clients. **Started** When the associated messaging engine is started, the WebSphere MQ client link is ready and enabled to receive inbound connections.

Additional Properties

Advanced properties

Advanced configurable properties, such as message reliability and broker queue names, for the WebSphere MQ client link.

WebSphere MQ links [Collection]

The WebSphere MQ link connects the messaging engine as a queue manager to WebSphere MQ, providing a bridge between the bus and a WebSphere MQ network.

To view this page in the console, click one of the following paths:

- **Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> [Related Items] WebSphere MQ links**
- **Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

If WebSphere MQ functionality has been disabled at any scope, an informational message indicating that WebSphere MQ has been disabled is displayed. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The name of the WebSphere MQ link, used for administrative purposes.

Description
An optional description for the WebSphere MQ link, for administrative purposes.

Local messaging engine
The local messaging engine that this WebSphere MQ link is hosted on.

Virtual queue manager name
The virtual queue manager name by which the local bus is to be known to a WebSphere MQ network. It is generally recommended that you set the virtual queue manager name to match the name of the local bus. As WebSphere MQ queue manager names can be no longer than 48 characters, you must ensure that the length of the local bus name does not exceed 48 characters.

Status
The runtime status of the WebSphere MQ link.

Status can take the following values:

Table 224. Status definitions. The first column of the table lists the status values of WebSphere MQ link. The second column contains a brief definition of the status.

Status	Meaning
Running	The WebSphere MQ link is running.
Stopped	The WebSphere MQ link is stopped.

Current outbound messages
The current total number of messages queued on the sender channel transmitter and link transmitters for this WebSphere MQ link.

Messages sent
The total number of messages sent, since the start of the messaging engine, on the sender channel transmitter to the foreign bus.

Messages received
The total number of messages received, since the start of the messaging engine, on the link receiver connections for this WebSphere MQ link.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.
Start	Start selected items. This button does not work if the selected WebSphere MQ link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.
Stop	Stop a selected WebSphere MQ link. You must first have selected the link to be stopped. You can choose the mode of stop action and the required state when the link has been stopped: **Stop mode:** **Force** Stop the link immediately. You should only use this mode if a quiesce action does not work. **Quiesce** Stop the link in a controlled manner. **Target state:** **Inactive** Stop the link and set its state to Inactive. If an application tries to use the link, the link is started again. **Stopped** Stop the link and set its state to Stopped. The link can only be started again by administrator action. This button does not work if the selected WebSphere MQ link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

WebSphere MQ link MQFAP inbound channel [Settings]

A channel that can be used in combination with the TCP Channel or other channels within the confines of WebSphere MQ support to facilitate communications between a WebSphere system integration bus and a WebSphere MQ client or queue manager.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server messaging] WebSphere MQ link inbound transports -> *chain_name* -> MQFAP inbound channel.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Transport channel name:

Specifies the unique name for a given layer in a network protocol stack.

Required	Yes
Data type	String

Discrimination weight:

Specifies the discrimination weight that is used to determine the order in which the channels obtain access to the incoming connection if the transport channels are shared amongst several transport chains. The transport channel with the lowest discrimination weight has the first opportunity to accept the incoming connection.

Required	No
Data type	Integer
Range	0 through 2147483647

Additional Properties

Custom properties

Specifies additional custom properties for this runtime component. Some components use custom configuration properties that can be defined here.

WebSphere MQ link receiver channel [Collection]

The receiver channel that receives messages from the gateway WebSphere MQ queue manager. The receiver channel communicates with a WebSphere MQ sender channel on the gateway queue manager, and converts MQ format messages to service integration bus messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Receiver channel.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

WebSphere MQ link receiver channel name

The name of the receiver channel for the WebSphere MQ link, used to receive messages from WebSphere MQ onto the bus.

Status

The runtime status of the receiver channel.

Buttons

Start	Start selected items. This button does not work if the selected WebSphere MQ link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.
Stop	Stop a selected channel on this link. You must first have selected the channel to be stopped. You can choose the mode of stop action and the required state when the channel has been stopped: **Stop mode:** **Force** Stop the channel immediately. You should only use this mode if a quiesce action does not work. **Quiesce** Stop the channel in a controlled manner. **Target state:** **Inactive** Stop the channel and set its state to Inactive. If an application tries to use the link, the channel is started again. **Stopped** Stop the channel and set its state to Stopped. The channel can only be started again by administrator action. This button does not work if the selected WebSphere MQ link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

WebSphere MQ link receiver channel [Settings]

The receiver channel that receives messages from the gateway WebSphere MQ queue manager. The receiver channel communicates with a WebSphere MQ sender channel on the gateway queue manager, and converts MQ format messages to service integration bus messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Receiver channel -> *channel_name*.

- “Configuration tab” on page 2209
- “Runtime tab” on page 2210

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WebSphere MQ link receiver channel name:

The name of the receiver channel for the WebSphere MQ link, used to receive messages from WebSphere MQ onto the bus.

This name must be the same as the name of the sender channel on WebSphere MQ.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type String

Inbound nonpersistent message reliability:

The acceptable reliability of message delivery for nonpersistent message flows from WebSphere MQ through this WebSphere MQ link, from Best effort to Reliable, in order of increasing reliability.

This reliability delivery option is assigned to all WebSphere MQ nonpersistent messages flowing over this receiver channel.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Best effort
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable
Messages are discarded when a messaging engine stops or fails.

Inbound persistent message reliability:

The acceptable reliability of message delivery for inbound persistent message flows from WebSphere MQ through this WebSphere MQ link, from Reliable to Assured, in order of increasing reliability.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Reliable
Messages might be discarded when a messaging engine fails.

Assured
Messages are not discarded.

Prefer queue points local to this link's messaging engine:

When this check box is selected, the link prefers to send inbound messages to available queue points of target destinations that are located on the messaging engine on which the link is hosted.

When this check box is not selected, or if no local queue point is available for a target destination, the link workload balances the messages across all available queue points of the target destination. By default the check box is selected.

This option is supported on links running on WebSphere Application Server Version 7.0 or later. If you are running on an earlier version, the default behavior of preferring local queue points is applied.

Required Yes
Data type Boolean

Initial state:

Whether the receiver channel is started or stopped when the associated WebSphere MQ link is first started. Until started, the channel is unavailable.

Dynamic updates to this property are effective on messaging engine restart or receiver channel creation. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime

property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Status:

The runtime status of the receiver channel.

This shows the overall status for all receiver channels for the link.

Required No

Data type drop-down list

Range

Inactive
The WebSphere MQ link is ready and enabled to create connections, but no connections have yet been established.

Stopped
The WebSphere MQ link is in a stopped state and cannot process any new requests for connections.

Multiple
The WebSphere MQ link has established several sessions with WebSphere MQ, and the sessions can be in different states: Starting, Running, Stopping, or Stopped.

Additional Properties

Receiver channel connections
The connections that exist on the receiver channel of the WebSphere MQ link.

Saved batch status
The runtime status of message batches for the receiver channel of the WebSphere MQ link.

WebSphere MQ link routing properties [Settings]

The routing properties for a link to a foreign bus that represents a WebSphere MQ network.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Foreign bus connections -> *foreign_bus_name* -> *link_name*.

You must create one of these routing definitions before you can create a WebSphere MQ link.

For information about setting up communication with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name by which the routing definition is known.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to the routing definition (virtual link) for administrative purposes.

Required	No
Data type	String

Inbound user ID:

The user ID applied to messages arriving from the foreign bus and used to authorize their access to destinations.

If you to want to authenticate messages entering this bus, this user ID replaces the user ID in the inbound messages. You can specify an inbound user ID if this bus is to assign its own user IDs for all messages received from the foreign bus. If you do not authenticate inbound messages, all messages can enter this bus.

Dynamic updates to this property are effective immediately.

Required	No
Data type	String

Outbound user ID:

The user ID applied to messages sent to the foreign bus.

The definitive check for access to the destination on the foreign bus takes place when the message enters the foreign bus. The check is against the permissions defined on the foreign bus by its administrator. The check is based on the user ID that is stored in the message. That user ID starts off as the user ID of the sender, but this bus might want to assign the same outbound user ID for all messages sent to the foreign bus.

Dynamic updates to this property are effective immediately.

Required	No
Data type	String

WebSphere MQ link sender channel [Collection]

This pane displays the sender channel that sends messages to the gateway queue manager. The sender channel communicates with a WebSphere MQ receiver channel on the gateway queue manager, and converts service integration bus messages to MQ format messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Sender channel.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

WebSphere MQ link sender channel name
The name of the sender channel for the WebSphere MQ link, used to send messages from the bus to WebSphere MQ.

Host name
The hostname or TCP/IP IP address for the gateway queue manager that is used to connect into the WebSphere MQ network.

Port The TCP/IP port number on which the gateway queue manager is listening for the WebSphere MQ link.

Status
The runtime status of the sender channel.

Buttons

Reset	Reset a selected channel sequence number to 1. If the channel is running, the reset request will take effect only when the channel has been stopped and restarted. This button does not work if the selected WebSphere MQ link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.
Start	Start selected items. This button does not work if the selected WebSphere MQ link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Stop	Stop a selected channel on this link. You must first have selected the channel to be stopped. You can choose the mode of stop action and the required state when the channel has been stopped: **Stop mode:** **Force** Stop the channel immediately. You should only use this mode if a quiesce action does not work. **Quiesce** Stop the channel in a controlled manner. **Target state:** **Inactive** Stop the channel and set its state to Inactive. If an application tries to use the link, the channel is started again. **Stopped** Stop the channel and set its state to Stopped. The channel can only be started again by administrator action. This button does not work if the selected WebSphere MQ link is running on an application server on which WebSphere MQ has been disabled. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

WebSphere MQ link sender channel [Settings]

This pane displays the sender channel that sends messages to the gateway queue manager. The sender channel communicates with a WebSphere MQ receiver channel on the gateway queue manager, and converts service integration bus messages to MQ format messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Sender channel -> *channel_name*.

- “Configuration tab”
- “Runtime tab” on page 2219

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WebSphere MQ link sender channel name:

The name of the sender channel for the WebSphere MQ link, used to send messages from the bus to WebSphere MQ.

This name must be the same as the name of the receiver channel on WebSphere MQ.

For information about choosing channel names, see the description of the Channel name (CHANNEL) property in the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

If the previous channel is unknown, restart is delayed until the uncertainty is resolved.

Required	No
Data type	String

Single Connection:

Use Single connection for the gateway queue manager that is used to connect into the WebSphere MQ network.

Select Single Connection to specify the host name and port for one target queue manager. If you have a high availability configuration, with one or more failover gateway queue managers available in the WebSphere MQ network in addition to the target queue manager, select **Multiple Connection Names List** instead.

If you do not specify any host names, either as a single connection or in a list of multiple connections, the gateway queue manager is assumed to be running on the same host as the messaging engine on which the WebSphere MQ link is defined.

Required	No
Data type	Radio button

Host name:

The hostname or TCP/IP IP address for the gateway queue manager that is used to connect into the WebSphere MQ network.

This is the host name or IP address of the target queue manager in the WebSphere MQ network.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	String

Port:

The TCP/IP port number on which the gateway queue manager is listening for the WebSphere MQ link.

This is the port number on which the gateway queue manager is listening for the inbound communication requests.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No

Data type	Integer
Range	0 through 65535
Default	1414

Multiple Connection Names List:

Select Multiple Connection Names List if you have one or more failover gateway queue managers available in the WebSphere MQ network in addition to the target queue manager. If no failover gateway queue manager is available, you can select Single Connection instead.

If you do not specify any host names, either as a single connection or in a list of multiple connections, the gateway queue manager is assumed to be running on the same host as the messaging engine on which the WebSphere MQ link is defined.

Required	No
Data type	Radio button

Connection Names List:

List of hostnames or TCP/IP IP address for the gateway queue manager that is used to connect into the WebSphere MQ network.

Specify a list of host names, or IP addresses, and ports for the active and standby WebSphere MQ queue managers in a high availability configuration. You may also use this field to specify a single host name and port.

Specify the host names, or IP addresses, and ports in the format `hostname1(nnnn),hostname2(nnnn)` separating each pair with a comma. Specify IP addresses in IPv4 format.

If you specify a port alone as an entry in the list, WebSphere Application Server uses the default host name `localhost` with the port. If you specify a host name alone as an entry in the list, WebSphere Application Server uses the default port number 1414 with the host name. For example, a connection names list consisting of the values `(1422),example.com` is interpreted as `localhost(1422),example.com(1414)`.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	Custom

Transport chain:

The type of channel chain used for communication with the foreign bus.

This must be one of the outbound transport chains in the application server, to be used when establishing a network connection to a WebSphere MQ queue manager receiver channel.

By default, this property can take one of the following values:

OutboundBasicMQLink
Used to establish connections with WebSphere MQ queue manager receiver channels.

OutboundSecureMQLink
Used to establish connections with WebSphere MQ queue manager receiver channels that have

been secured by using SSL. The SSL configuration used is taken from the default SSL repertoire for the application server being used to contact the queue manager.

You can also choose to specify another outbound transport chain that you have defined separately on the TransportChannelService object of the application server used for WebSphere MQ interoperation.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Modified transport chains are visible only after a server restart.

Required	Yes
Data type	Custom

Disconnect interval:

The time in seconds for which the sender channel waits for new messages to arrive on the transmission queue after sending a batch of messages. The channel disconnects after this interval, and must be restarted manually or by triggering.

The default value is a reasonable interval. Change this value only if you understand the implications for performance, and you need a different value for the requirements of the traffic flowing down your channels.

Performance is affected by the value specified for the disconnect interval. A very low value (a few seconds) can cause an unacceptable amount of processing in constantly starting up the channel. A very large value (more than an hour) might mean that system resources are unnecessarily held up.

If you want your channels to be active only when there are messages for them to transmit, you should set the disconnect interval to a fairly low value. Note that the default setting is quite high and so is not recommended for channels where this level of control is required. Because it is difficult to interrupt the receiving channel, the most economical option is to have the channel automatically disconnect and reconnect as the workload demands. For most channels, the appropriate setting of the disconnect interval can be established heuristically.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	Integer
Range	0 through 999999 A value of 0 (zero) means never disconnect; the channel waits indefinitely for messages.

Short retry count:

The maximum number of times that the sender channel tries to restart after a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then the long retry mechanism is invoked.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	Integer
Range	0 through 999999999

Short retry interval:

The number of seconds between attempts by the sender channel to restart after a communication or partner failure.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	Integer
Range	0 through 999999999

Long retry count:

The maximum number of times that the sender channel tries to restart after the short retry mechanism did not recover from a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then an error is logged and the channel is stopped.

For more information about about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	Long
Range	0 through 999999999

Long retry interval:

The number of seconds between attempts by the sender channel to restart after the short retry mechanism did not recover from a communication or partner failure.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No

Data type Long
Range 0 through 999999999

Initial state:

Whether the sender channel is started or stopped when the associated WebSphere MQ link is first started. Until started, the channel is unavailable.

Dynamic updates to this property are effective on messaging engine restart or sender channel creation. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WebSphere MQ link sender channel name:

The name of the sender channel for the WebSphere MQ link, used to send messages from the bus to WebSphere MQ.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

If the previous channel is unknown, restart is delayed until the uncertainty is resolved.

Virtual queue manager name:

The virtual queue manager name by which the local bus is to be known to a WebSphere MQ network. It is generally recommended that you set the virtual queue manager name to match the name of the local bus. As WebSphere MQ queue manager names can be no longer than 48 characters, you must ensure that the length of the local bus name does not exceed 48 characters.

The bus appears to WebSphere MQ as if it is a WebSphere MQ queue manager or queue-sharing group. This field gives the queue manager name that WebSphere MQ uses to address the virtual queue manager represented by this bus.

Required	No
Data type	String

I/P address:

The TCP/IP IP address of the host on which the target queue manager runs.

Required	No
Data type	String

Status:

The runtime status of the sender channel.

Required	No
Data type	drop-down list

Range

Inactive
The WebSphere MQ link is ready and enabled to create connections, but no connections have yet been established.

Starting
The WebSphere MQ link has received a connection request, and is in the process of starting this connection.

Binding
The WebSphere MQ link is performing channel negotiation and is not yet ready to transfer messages.

Initializing
The WebSphere MQ link is initializing the session for a connection and is not yet ready to transfer messages.

Retrying
The WebSphere MQ link is retrying a failed connection.

Standby
The channel is being used for standby purposes. Messages can be transferred only when the channel is active.

Running
The WebSphere MQ link has an active session with WebSphere MQ, and data can be flowing between WebSphere MQ and the WebSphere MQ link.

Stopping
The WebSphere MQ link, or a connection with that link, is in the process of being stopped.

Paused
The channel is waiting for the message-retry interval to finish.

Stopped
The WebSphere MQ link is in a stopped state and cannot process any new requests for connections.

Multiple
The WebSphere MQ link has established several sessions with WebSphere MQ, and the sessions can be in different states: Starting, Running, Stopping, or Stopped.

Host name:

The hostname or TCP/IP IP address for the gateway queue manager that is used to connect into the WebSphere MQ network.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Port:

The TCP/IP port number on which the gateway queue manager is listening for the WebSphere MQ link.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Transport chain:

The type of channel chain used for communication with the foreign bus.

This must be one of the outbound transport chains in the application server, to be used when establishing a network connection to a WebSphere MQ queue manager receiver channel. For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Modified transport chains are visible only after a server restart.

Disconnect interval:

The time in seconds for which the sender channel waits for new messages to arrive on the transmission queue after sending a batch of messages. The channel disconnects after this interval, and must be restarted manually or by triggering.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Short retry count:

The maximum number of times that the sender channel tries to restart after a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then the long retry mechanism is invoked.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Short retry interval:

The number of seconds between attempts by the sender channel to restart after a communication or partner failure.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Long retry count:

The maximum number of times that the sender channel tries to restart after the short retry mechanism did not recover from a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then an error is logged and the channel is stopped.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Long retry interval:

The number of seconds between attempts by the sender channel to restart after the short retry mechanism did not recover from a communication or partner failure.

For more information, see the description of this property for the Configuration tab.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Batch size:

The maximum number of messages that a batch can contain.

Required	No
Data type	String

Channel start time:

The time at which the channel was last started.

Required	No
Data type	String

Channel start date:

The date on which the channel was last started.

Required	No
Data type	String

Heartbeat interval:

The negotiated time, in seconds, between heartbeat flows passed from the WebSphere MQ link sender channel to the WebSphere MQ receiver channel when there are no messages on the transmission queue.

Required	No
Data type	Integer

Sequence wrap:

The value at which message sequence numbers wrap to start again at 1.

This is the highest number the message sequence number reaches before it restarts at 1. The default value is 999999999.

For more information about choosing the value for this property, see the description of the Sequence Number wrap (SEQWRAP) property in the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type Long
Range 100 through 999999999

Maximum message length:

The negotiated maximum message length, in bytes, that can be transmitted on the channel. This value is compared with the value from the partner WebSphere MQ receiver channel, and the maximum message length used is the lower of the two values.

Required No
Data type Integer

Stop requested:

Whether or not a manual stop request has been made for the sender channel.

To make a stop request, use the buttons provided on the WebSphere MQ link sender channel collection panel.

Required No
Data type drop-down list
Range

true The channel is in doubt about which messages have been committed by WebSphere MQ for the unit of work that it has sent.

false The channel is not in doubt about which messages have been committed by WebSphere MQ.

Current LUWID:

The current logical unit-of-work identifier for the message on the channel.

Indoubt channel problems are usually resolved automatically. Sequence number and logical unit of work ID (LUWID) records are kept to help resolve indoubt channel problems when communication has been re-established. For information resolving indoubt channels, see the *Intercommunication* section of the WebSphere MQ information center.

Required No
Data type String

Current sequence number:

The current sequence number for the message on the channel.

Indoubt channel problems are usually resolved automatically. Sequence number and logical unit of work ID (LUWID) records are kept to help resolve indoubt channel problems when communication has been re-established. For information resolving indoubt channels, see the *Intercommunication* section of the WebSphere MQ information center.

Required No
Data type String

In doubt:

Whether the message on the channel is in an indoubt state.

Problems with a channel that is in an indoubt state are usually resolved automatically. Sequence number and logical unit of work ID (LUWID) records are kept to help resolve problems with an indoubt channel when communication has been re-established. For information about resolving indoubt channels, see the *Intercommunication* section of the WebSphere MQ information center.

Required No
Data type drop-down list
Range

true The channel is in doubt about which messages have been committed by WebSphere MQ for the unit of work that it has sent.

false The channel is not in doubt about which messages have been committed by WebSphere MQ.

Last LUWID:

The last logical unit-of-work identifier for a message on the channel.

To determine the last-committed logical unit of work ID (LUWID) for the channel, compare this value to the last-committed LUWID for the receiving side of the channel.

- If the two LUWIDs are the same, the receiving side has committed the unit of work that the sender considers to be in doubt. The sending side can now remove the indoubt messages from the transmission queue and re-enable it.
- If the two LUWIDs are different, the receiving side has not committed the unit of work that the sender considers to be in doubt. The sending side needs to retain the indoubt messages on the transmission queue and re-send them.

Required No
Data type String

Last sequence number:

The last sequence number for a message on the channel.

WebSphere MQ channels cannot initialize successfully if the channel negotiation detects that the message sequence number is different at each end. You might have to reset this manually.

To be effective, the sequence number must be reset in both the sender and the receiver channel definitions. The starting sequence number is not negotiated when a channel starts up, nor is there a default provided.

Required	No
Data type	String

Messages in current batch:

The number of messages in the current batch on the channel.

Required	No
Data type	String

Number of batches sent:

The number of batches that have been sent on the channel.

Required	No
Data type	String

Number of messages sent:

The number of messages that have been sent on the channel.

Required	No
Data type	String

Buffers sent:

The number of message buffers sent.

Required	No
Data type	String

Buffers received:

The number of message buffers received.

Required	No
Data type	String

Bytes sent:

The number of bytes sent.

Required	No
Data type	String

Bytes received:

The number of bytes received.

Required No
Data type String

Last message send time:

The time at which the last message was sent on the channel.

Required No
Data type String

Last message send date:

The date on which the last message was sent on the channel.

Required No
Data type String

Remaining short retry starts:

The remaining number of short retry attempts that can be used to start the sender channel. If the count of remaining retries reaches zero, and the channel has not restarted, then the long retry mechanism is invoked.

Required No
Data type String

Remaining long retry starts:

The remaining number of long retry attempts that can be used to start the sender channel. If the count of remaining retries reaches zero, and the channel has not restarted, then an error is logged and the channel is stopped.

Required No
Data type String

Nonpersistent message speed:

The class of service for nonpersistent messages on the sender channel.

Required No
Data type drop-down list
Range

Fast Nonpersistent messages can be lost if there is a transmission failure or if the channel stops when the messages are in transit.

Normal Nonpersistent messages are not lost if there is a transmission failure or if the channel stops when the messages are in transit.

Additional Properties

Saved batch status

The runtime status of message batches for the sender channel of the WebSphere MQ link. You can choose to commit or roll back each batch.

WebSphere MQ link [Settings]

The WebSphere MQ link connects the messaging engine as a queue manager to WebSphere MQ, providing a bridge between the bus and a WebSphere MQ network.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name*

For more information about choosing settings, see the *Intercommunication* section of the WebSphere MQ information center.

If WebSphere MQ functionality has been disabled at any scope, an informational message indicating that WebSphere MQ has been disabled is displayed. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

- “Configuration tab”
- “Runtime tab” on page 2233

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the WebSphere MQ link, used for administrative purposes.

Required	Yes
Data type	String

UUID:

The universal unique identifier assigned by the system to this WebSphere MQ link for administrative purposes.

Required	No
Data type	String

Description:

An optional description for the WebSphere MQ link, for administrative purposes.

Dynamic updates to this property are effective immediately.

Required No
Data type Text area

Foreign bus connection name:

The foreign bus to which this link connects.

If you do not see the foreign bus connection name that you expected to see, you must create a new foreign bus connection.

Required Yes
Data type drop-down list

Local messaging engine:

The local messaging engine that this WebSphere MQ link is hosted on.

Required Yes
Data type drop-down list

Virtual queue manager name:

The virtual queue manager name by which the local bus is to be known to a WebSphere MQ network. It is generally recommended that you set the virtual queue manager name to match the name of the local bus. As WebSphere MQ queue manager names can be no longer than 48 characters, you must ensure that the length of the local bus name does not exceed 48 characters.

The bus appears to WebSphere MQ as if it is a WebSphere MQ queue manager or queue-sharing group. This field gives the queue manager name that WebSphere MQ uses to address the virtual queue manager represented by this bus. Any value that you specify must meet the following criteria:

- It must contain between 1 and 48 characters.
- It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

Required Yes
Data type String

Batch size:

The maximum number of messages that can be sent through a channel before taking a checkpoint.

The batch size does not affect the way the sender and receiver channels for this link transfer messages; messages are always transferred individually, but are committed or backed out as a batch.

For information about choosing the batch size, see the description of the Batch size (BATCHSZ) property in the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type Integer

Range 1 through 9999

Maximum message size:

The maximum message length, in bytes, that can be transmitted on any channel for the WebSphere MQ link. This is compared with the value for the corresponding partner WebSphere MQ channel and the actual maximum used is the lower of the two values.

For more information about choosing a value for the maximum message size, see the description of the Maximum message length (**MAXMSGL**) property in the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type Integer
Range 0 through 104857600

Heartbeat interval:

The negotiated time, in seconds, between heartbeat flows passed from the WebSphere MQ link sender channel to the WebSphere MQ receiver channel when there are no messages on the transmission point being served by the WebSphere MQ link sender channel.

Heartbeats give the receiving channel the opportunity to quiesce the channel connection.

For more information about choosing a value for this property, see the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type Integer
Range 0 through 999999

Sequence wrap:

The value at which message sequence numbers wrap to start again at 1.

For more information about choosing a value for this property, see the description of the Sequence Numberwrap (**SEQWRAP**) property in the *Intercommunication* section of the WebSphere MQ information center.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No

Data type	Long
Range	100 through 999999999

Adoptable:

Whether or not a running instance of a WebSphere MQ link receiver channel (associated with this MQ link) should be adopted or not. In the event of a communications failure, it is possible for a running instance of a WebSphere MQ link receiver channel to be left waiting for messages. When communication is re-established, and the partner WebSphere MQ sender channel next attempts to establish a session with the WebSphere MQ link receiver channel, the request will fail as there is already a running instance of the WebSphere MQ link receiver channel that believes it is in session with the partner WebSphere MQ sender channel. You can overcome this problem by selecting this option, which causes the already running instance of the WebSphere MQ link receiver channel to be stopped and a new instance to be started.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	Boolean

Exception destination:

The destination for an inbound message when the WebSphere MQ link cannot deliver the message to its target bus destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist.

Select a radio button as required to configure the exception destination that this WebSphere MQ link uses:

- Select **None** to specify that the WebSphere MQ link does not use an exception destination. Undeliverable messages are not rerouted to an exception destination and can block the processing of other messages waiting for delivery through that link to the same bus. This option can be used to preserve message ordering.
- Select **System** to use the default exception destination. Messages that cannot be delivered to the bus destination are rerouted to the system default exception destination for the messaging engine that this link is assigned to: _SYSTEM.Exception.Destination*messaging_engine_name*.
- Select **Specify** and enter an exception destination to use the exception destination specified here. If the WebSphere MQ link cannot use this exception destination, it uses the system exception destination.

Required	No
Data type	Custom
Default	System

Initial state:

Whether the WebSphere MQ link is started or stopped when the hosting messaging engine is first started. Until it is started, the WebSphere MQ link is unavailable.

Dynamic updates to this property are effective when the messaging engine restarts. Use the Runtime tab to check the current state.

Required	No
Data type	drop-down list

Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

Nonpersistent message speed:

The class of service for nonpersistent messages on channels of this WebSphere MQ link.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Fast Nonpersistent messages can be lost if there is a transmission failure or if the channel stops when the messages are in transit.

Normal
Nonpersistent messages are not lost if there is a transmission failure or if the channel stops when the messages are in transit.

Additional Properties

Publish/subscribe broker profiles
Profiles used to define the topic mappings and transactionality for publishing and receiving (by subscription) topics across the publish/subscribe bridge between WebSphere Application Server and a WebSphere MQ network.

Receiver channel
The receiver channel that receives messages from the gateway queue manager. The receiver channel communicates with a WebSphere MQ sender channel on the gateway queue manager, and converts MQ format messages to service integration bus messages.

Sender channel
The sender channel that sends messages to the gateway queue manager. The sender channel communicates with a WebSphere MQ receiver channel on the gateway queue manager, and converts service integration bus messages to MQ format messages.

Sender channel transmitters
The sender channel transmitters for the queueing of messages across the WebSphere MQ link.

Related Items

Foreign bus connection
The associated foreign bus for this WebSphere MQ link.

Link transmitters

For applications that use point-to-point messaging, there is one link transmission message point located on each messaging engine in the source bus. For applications that use publish/subscribe messaging, there is one link transmission message point located on each topic space in the source bus. The link transmitter acts as a transmission queue where produced messages are persisted before transmission across the inter-bus link to the foreign bus.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Status:

The runtime status of the WebSphere MQ link.

Required	No
Data type	drop-down list

Range

Inactive
The WebSphere MQ link is ready and enabled to create connections, but no connections have yet been established.

Starting
The WebSphere MQ link has received a connection request, and is in the process of starting this connection.

Binding
The WebSphere MQ link is performing channel negotiation and is not yet ready to transfer messages.

Initializing
The WebSphere MQ link is initializing the session for a connection and is not yet ready to transfer messages.

Retrying
The WebSphere MQ link is retrying a failed connection.

Standby
The channel is being used for standby purposes. Messages can be transferred only when the channel is active.

Running
The WebSphere MQ link has an active session with WebSphere MQ, and data can be flowing between WebSphere MQ and the WebSphere MQ link.

Stopping
The WebSphere MQ link, or a connection with that link, is in the process of being stopped.

Paused
The channel is waiting for the message-retry interval to finish.

Stopped
The WebSphere MQ link is in a stopped state and cannot process any new requests for connections.

Multiple
The WebSphere MQ link has established several sessions with WebSphere MQ, and the sessions can be in different states: Starting, Running, Stopping, or Stopped.

Additional Properties

Sender channel transmitters
The sender channel transmitters for the queueing of messages across the WebSphere MQ link.

Related Items

Link transmitters
For applications that use point-to-point messaging, there is one link transmission message point

located on each messaging engine in the source bus. For applications that use publish/subscribe messaging, there is one link transmission message point located on each topic space in the source bus. The link transmitter acts as a transmission queue where produced messages are persisted before transmission across the inter-bus link to the foreign bus.

Mediation execution points [Collection]

Mediation execution points for the processing of messages from mediation message points that are on a WebSphere MQ server.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Remote message points] WebSphere MQ Mediation Execution points.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

Status
The runtime status of the mediation point.

Buttons

Start	Start selected items.
Stop	Stop selected items.

Mediation points [Collection]

Bus member (server or cluster) where the mediations for the destination run.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message points] WebSphere MQ Mediation Execution points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Buttons

Start	Start selected items.
Stop	Stop selected items.

Mediation points [Settings]

Bus member (server or cluster) where the mediations for the destination run.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Remote message points] WebSphere MQ Mediation Execution points -> *mediation_name* .

- “Configuration tab”
- “Runtime tab”

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required No
Data type String

UUID:

The universal unique identifier assigned by the system to this message point for administrative purposes.

Required No
Data type String

Initial state:

Whether the mediation point is started or stopped when the hosting messaging engine is first started. Until started, the mediation point is unavailable.

Required No
Data type drop-down list
Range

Started
When the associated messaging engine is started, the mediation is started and is available to process messages.

Stopped
When the associated messaging engine is started, the mediation is stopped and is not available to process messages.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime

property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required	No
Data type	String

Run-time ID:

The internal runtime identifier assigned to this message point.

Required	No
Data type	String

Status:

The runtime status of the mediation point.

Required	No
Data type	drop-down list
Range	**Waiting** The mediation is waiting to start. This might be because the application server is not yet open for e-business, or because a previous instance of the mediation has not yet been deleted. **Started** The mediation is started and is available to process messages. **Stopping** The mediation is in the process of stopping. **Stopped** The mediation is stopped. The reason why the mediation is stopped is shown in the Reason attribute. **Deleting** The mediation is in the process of being deleted.

WebSphere MQ mediation points [Collection]

A WebSphere MQ mediation point is a location from which a mediation takes messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message points] WebSphere MQ Mediation points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

WebSphere MQ mediation points [Settings]

A WebSphere MQ mediation point is a location from which a mediation takes messages.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message points] WebSphere MQ Mediation points -> *mediation_name*** .

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this message point for administrative purposes.

Required	No
Data type	String

Target UUID:

The UUID of the bus destination for which this is a message point.

Required	No
Data type	String

WebSphere MQ queue name :

The WebSphere MQ queue name as defined in WebSphere MQ.

Required	Yes
Data type	String
Data type	String

Inbound nonpersistent reliability :

The level of reliability to apply when a nonpersistent WebSphere MQ message is received.

Required
No

Data type
drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Inbound persistent reliability :

The level of reliability to apply when a persistent WebSphere MQ message is received.

Required
No

Data type
drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Include an RFH2 message header when sending messages to WebSphere MQ :

If selected, messages sent to WebSphere MQ will include an RFH2 header. The RFH2 header stores additional information to that which is stored in the WebSphere MQ message header.

Required	No
Data type	Boolean
Data type	Boolean

WebSphere MQ queue points [Collection]

A WebSphere MQ queue point is used when sending messages to and receiving messages from a WebSphere MQ queue.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message points] WebSphere MQ Queue points**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Identifier
The system-generated name by which this message point is known.

WebSphere MQ queue points [Settings]

A WebSphere MQ queue point is used when sending messages to and receiving messages from a WebSphere MQ queue.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> Runtime > **[Message points] WebSphere MQ Queue points -> *queue_point_name***.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Identifier:

The system-generated name by which this message point is known.

Required	No
Data type	String

UUID:

The universal unique identifier assigned by the system to this message point for administrative purposes.

Required	No
Data type	String

Target UUID:

The UUID of the bus destination for which this is a message point.

Required	No
Data type	String

WebSphere MQ queue name :

The WebSphere MQ queue name as defined in WebSphere MQ.

Required	Yes
Data type	String

Inbound nonpersistent reliability :

The level of reliability to apply when a nonpersistent WebSphere MQ message is received.

The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For nonpersistent WebSphere MQ messages received, you will probably choose one of the nonpersistent service integration qualities of service, that is reliable nonpersistent, best effort nonpersistent or express nonpersistent. However, you can choose any of the five possible service integration qualities of service:

Best effort nonpersistent

Express nonpersistent

Reliable nonpersistent

Reliable persistent

Assured persistent

For more information, see ../ae/rjc0014_.dita.

Required No

Data type drop-down list

Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Inbound persistent reliability :

The level of reliability to apply when a persistent WebSphere MQ message is received.

The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For persistent WebSphere MQ messages received, you will probably choose one of the persistent service integration qualities of service, that is either assured persistent or reliable persistent. However, you can choose any of the five possible service integration qualities of service:

Best effort nonpersistent

Express nonpersistent

Reliable nonpersistent

Reliable persistent

Assured persistent

For more information, see ../ae/rjc0014_.dita.

Required No
Data type drop-down list
Range

Best effort nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express nonpersistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable nonpersistent
Messages are discarded when a messaging engine stops or fails.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Assured persistent
Messages are not discarded.

Include an RFH2 message header when sending messages to WebSphere MQ :

If selected, messages sent to WebSphere MQ will include an RFH2 header. The RFH2 header stores additional information to that which is stored in the WebSphere MQ message header.

Required No
Data type Boolean

WebSphere MQ link receiver channel connections [Collection]

A connection that exists on the receiver channel of the WebSphere MQ link.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Receiver channel -> *channel_name* -> Runtime > **Receiver channel connections**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

WebSphere MQ Queue Manager name
The name of the WebSphere MQ queue manager on which the WebSphere MQ sender channel, that is connected to this MQ link receiver channel connection instance, is running.

IP address
The IP address of the queue manager for the WebSphere MQ link receiver channel connection.

Status
The runtime status of the WebSphere MQ link receiver channel connection.

WebSphere MQ link receiver channel connections [Settings]

A connection that exists on the receiver channel of the WebSphere MQ link.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Messaging engines -> *engine_name* -> [Additional properties] WebSphere MQ links -> *link_name* -> [Additional Properties] Receiver channel -> *channel_name* -> Runtime > **Receiver channel connections -> *channel_connection_name***.

Runtime tab

The Runtime tab shows runtime properties and current runtime state for this MQ link receiver channel connection. These properties are not preserved when the current runtime environment is stopped.

General Properties

WebSphere MQ link receiver channel name:

The name of the receiver channel for the WebSphere MQ link, used to receive messages from WebSphere MQ onto the bus.

This name must be the same as the name of the sender channel on WebSphere MQ.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	String

WebSphere MQ Queue Manager name:

The name of the WebSphere MQ queue manager on which the WebSphere MQ sender channel, that is connected to this MQ link receiver channel connection instance, is running.

Required	No
Data type	String

IP address:

The IP address of the queue manager for the WebSphere MQ link receiver channel connection.

Required	No
Data type	String

Status:

The runtime status of the WebSphere MQ link receiver channel connection.

Required No
Data type drop-down list
Range

Inactive
The WebSphere MQ link is ready and enabled to create connections, but no connections have yet been established.

Starting
The WebSphere MQ link has received a connection request, and is in the process of starting this connection.

Binding
The WebSphere MQ link is performing channel negotiation and is not yet ready to transfer messages.

Initializing
The WebSphere MQ link is initializing the session for a connection and is not yet ready to transfer messages.

Retrying
The WebSphere MQ link is retrying a failed connection.

Standby
The channel is being used for standby purposes. Messages can be transferred only when the channel is active.

Running
The WebSphere MQ link has an active session with WebSphere MQ, and data can be flowing between WebSphere MQ and the WebSphere MQ link.

Stopping
The WebSphere MQ link, or a connection with that link, is in the process of being stopped.

Paused
The channel is waiting for the message-retry interval to finish.

Stopped
The WebSphere MQ link is in a stopped state and cannot process any new requests for connections.

Multiple
The WebSphere MQ link has established several sessions with WebSphere MQ, and the sessions can be in different states: Starting, Running, Stopping, or Stopped.

Batch size:

The maximum number of messages that a batch can contain.

Required	No
Data type	String

Channel start time:

The time at which the channel was last started.

Required	No
Data type	String

Channel start date:

The date on which the channel was last started.

Required	No
Data type	String

Heartbeat interval:

The negotiated time, in seconds, between heartbeat flows passed from the WebSphere MQ sender channel to the WebSphere MQ link receiver channel when there are no messages on the transmission queue being served by the WebSphere MQ sender channel.

Required	No
Data type	String

Sequence wrap:

The value at which message sequence numbers wrap to start again at 1.

This is the highest number the message sequence number reaches before it restarts at 1. The default value is 999999999.

For information about how to choose an appropriate value for this property, see the description of the Sequence Numberwrap (SEQWRAP) property in the *WebSphere MQ Intercommunication* book.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required	No
Data type	Long
Range	100 through 999999999

Adoptable:

Whether or not a running instance of a WebSphere MQ link receiver channel (associated with this MQ link) should be adopted or not. In the event of a communications failure, it is possible for a running instance of a WebSphere MQ link receiver channel to be left waiting for messages. When communication is re-established, and the partner WebSphere MQ sender channel next attempts to establish a session with the WebSphere MQ link receiver channel, the request will fail as there is already a running instance

of the WebSphere MQ link receiver channel that believes it is in session with the partner WebSphere MQ sender channel. You can overcome this problem by selecting this option, which causes the already running instance of the WebSphere MQ link receiver channel to be stopped and a new instance to be started.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

true

false

Maximum message length:

The maximum message length, in bytes, that can be transmitted on the channel. This is compared with the value from the partner Websphere MQ sender channel and the actual maximum used is the lower of the two values.

Required No
Data type String

Stop requested:

Whether or not a manual stop request has been made for the receiver channel.

Required No
Data type drop-down list
Range

true The channel is in doubt about which messages have been committed by WebSphere MQ for the unit of work that it has sent.

false The channel is not in doubt about which messages have been committed by WebSphere MQ.

Current LUWID:

The current logical unit-of-work identifier of the WebSphere MQ link receiver channel connection.

Required No
Data type String

Current sequence number:

The current sequence number for the message on the channel.

You can reset the sequence number of an MQLinkReceiver channel by resetting the sequence number of the partner WebSphere MQ sender channel. For details of how to reset the sequence number of a WebSphere MQ sender channel, see the *WebSphere MQ Intercommunications Guide* section of the WebSphere MQ library.

Required No

Data type String

Last LUWID:

The last logical unit-of-work identifier for a message on the channel.

Required No
Data type String

Last Sequence number:

The last sequence number for a message on the channel.

Required No
Data type String

Messages in current batch:

The number of messages in the current batch on the channel.

Required No
Data type String

Number of batches received:

The number of batches received on the channel.

Required No
Data type String

Number of messages received:

The number of messages received on the channel.

Required No
Data type String

Buffers sent:

The number of buffers sent on the channel.

Required No
Data type String

Buffers received:

The number of buffers received on the channel.

Required No
Data type String

Bytes sent:

The number of bytes sent on the channel.

Required	No
Data type	String

Bytes received:

The number of bytes received on the channel.

Required	No
Data type	String

Last message receive time:

The time at which the last message was received on the receiver channel connection.

Required	No
Data type	String

Last message receive date:

The date on which the last message was received on the receiver channel connection.

Required	No
Data type	String

nonpersistent message speed:

The class of service for nonpersistent messages on the receiver channel.

Required	No
Data type	drop-down list
Range	**Fast** Nonpersistent messages can be lost if there is a transmission failure or if the channel stops when the messages are in transit.
	Normal Nonpersistent messages are not lost if there is a transmission failure or if the channel stops when the messages are in transit.

Inbound nonpersistent message reliability:

The acceptable reliability of message delivery for nonpersistent message flows from WebSphere MQ through this WebSphere MQ link, from Best effort to Reliable, in order of increasing reliability.

This reliability delivery option is assigned to all WebSphere MQ nonpersistent messages flowing over this receiver channel.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Best effort
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable
Messages are discarded when a messaging engine stops or fails.

Inbound persistent message reliability:

The acceptable reliability of message delivery for inbound persistent message flows from WebSphere MQ through this WebSphere MQ link, from Reliable to Assured, in order of increasing reliability.

Dynamic updates to this property are effective when the channel restarts. Use the Runtime tab to check the current state.

Required No
Data type drop-down list
Range

Reliable
Messages might be discarded when a messaging engine fails.

Assured
Messages are not discarded.

WebSphere MQ server bus member [Settings]

A WebSphere MQ server bus member is used for assigning queue points and mediation points to WebSphere MQ queues.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Topology] Bus members -> *member_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The administrative name of the WebSphere MQ server bus member.

Required	No
Data type	String

UUID:

The universal unique identifier is assigned automatically, for administrative purposes, when you create a new WebSphere MQ server definition.

Required	No
Data type	String

Connection settings

Virtual queue manager name:

When sending messages to WebSphere MQ, the WebSphere MQ gateway queue manager sees the bus as a remote queue manager. The virtual queue manager name is passed to WebSphere MQ as the name of the remote queue manager.

The default value is the name of the Service Integration bus. If this bus name is not a valid name for a WebSphere MQ queue manager, or if another WebSphere MQ queue manager already has the same name, replace the default value with a unique, valid name for a WebSphere MQ queue manager.

- It must contain between 1 and 48 characters.
- It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

Required	Yes
Data type	String

Override WebSphere MQ Server connection properties:

When you select this checkbox, the bus-specific properties override the WebSphere MQ server connection properties. To reset the connection settings to the WebSphere MQ server default values, deselect this checkbox and apply the change.

Required	No
Data type	Boolean

WebSphere MQ Host:

The name of the WebSphere MQ server's host.

This is either the symbolic name or the IP address of the host to which a connection is established for communicating with the queue manager or queue-sharing group that this WebSphere MQ server represents. The value is a string and must be one of the following:

- Symbolic host name
- IPv4 address

- IPv6 address

Required No
Data type String

Port:

The WebSphere MQ server port. This value is the TCP/IP port number on which the queue manager or queue-sharing group that this WebSphere MQ server represents listens. This value is used when attempting to establish client transport mode connections to WebSphere MQ. The default value is 1414

Required No
Data type Integer

Transport chain name:

The name of the WebSphere MQ server transport chain. This is the transport chain that is used to establish an outbound network connection to the WebSphere MQ server.

Required No
Data type drop-down list

WebSphere MQ channel:

The name of the WebSphere MQ server's channel. This value is the WebSphere MQ client channel name to use when connecting to the queue manager or queue-sharing group that this WebSphere MQ server represents. The default value is SYSTEM.DEF.SVRCONN

Required No
Data type String

Messaging authentication alias:

The name of WebSphere MQ server's authentication alias. This is the name that is used for exchanging messages with WebSphere MQ.

Required No
Data type drop-down list

Trust user identifiers received in messages:

The WebSphere MQ server's trust user identifier setting. This value determines whether user identifiers that are received in messages from WebSphere MQ are propagated into the message (that is, whether user identifiers that are received as part of message data are used in the service integration bus).

If you select this checkbox, user identifiers that are received as part of WebSphere MQ messages are propagated into messages and used for security purposes within the bus. If you do not select this checkbox, the user identifier is overwritten with the WebSphere MQ server name.

Required No
Data type Boolean

Test connection:

After you have configured the Connection properties, click this button to test the connection to WebSphere MQ.

Message points

WebSphere MQ mediation points
Displays a list of WebSphere MQ mediation points assigned to this WebSphere MQ server bus member.

WebSphere MQ queue points
Displays a list of WebSphere MQ queue points assigned to this WebSphere MQ server bus member.

Related Items

WebSphere MQ server
The WebSphere MQ server that was used to create this bus member.

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

WebSphere MQ servers [Collection]

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere MQ servers.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

If WebSphere MQ functionality has been disabled at any scope, an informational message indicating that WebSphere MQ has been disabled is displayed. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Name The administrative name of the WebSphere MQ Server.

Description
An optional description for the WebSphere MQ server, for administrative purposes.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Test connection	Test the connection to WebSphere MQ. In a single server environment, any attempt to test the connection fails if WebSphere MQ functionality has been disabled. In a network deployment environment, any attempt to test the connection fails if WebSphere MQ functionality has been disabled at either the cell or deployment manager node scope. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

WebSphere MQ server [Settings]

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. Use this panel to view or modify the details of a WebSphere MQ server definition.

To view this page in the console, click one of the following paths:

- **Servers -> New server**, choose a server type of “WebSphere MQ server”, then click **Next**.
- **Servers -> Server Types -> WebSphere MQ servers -> New**
- **Servers -> Server Types -> WebSphere MQ servers -> *server_name***
- **Service integration -> Buses -> *bus_name* -> [Topology] Bus members -> *member_name* -> [Related Items] WebSphere MQ server**

If WebSphere MQ functionality has been disabled at any scope, an informational message indicating that WebSphere MQ has been disabled is displayed. In a single server environment this informational message is only displayed when the server is restarted after WebSphere MQ functionality has been disabled. In a network deployment environment the informational message is displayed immediately. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The administrative name of the WebSphere MQ Server.

The name that you use for this WebSphere MQ server definition must be unique.

Required Yes
Data type String

UUID:

The universal unique identifier assigned by the system to this WebSphere MQ server for administrative purposes.

This identifier is assigned automatically when you create a new WebSphere MQ server definition.

Required No
Data type String

Description:

An optional description for the WebSphere MQ server, for administrative purposes.

Required No
Data type Text area

WebSphere MQ server name:

The name of the WebSphere MQ queue manager or queue sharing group, as defined in WebSphere MQ.

Required Yes
Data type String

Server Type:

A server can be either a queue manager or queue sharing group. Select the appropriate type for the server you want to establish a connection to.

Required No
Data type Radio button
Range

Queue manager
A WebSphere MQ queue manager on the WebSphere MQ network with which WebSphere Application Server applications undertake messaging.

Queue sharing group
A WebSphere MQ for z/OS queue-sharing group on the WebSphere MQ network with which WebSphere Application Server applications undertake messaging.

Use bindings transport mode if available:

When selected, bindings transport mode will be used in preference to client transport mode when establishing connections to the WebSphere MQ server. Otherwise client transport mode will always be used.

To connect to a WebSphere MQ queue manager or queue-sharing group in bindings mode, WebSphere Application Server needs to know where to load native libraries from. This information is stored in the **Native library path** property of the WebSphere MQ messaging provider. If you want to use a direct binding to WebSphere MQ, rather than a TCP/IP network connection, select this option and configure the **Native library path** property as described in “Configuring the WebSphere MQ messaging provider with native libraries information” on page 723.

Note: If you are using Resource Access Control Facility (RACF) as the security manager on your WebSphere MQ for z/OS system, and using bindings transport mode, you must specify in uppercase characters the user names and passwords for authentication aliases. If you are using RACF and client transport mode, you can specify the user names and passwords in either upper or lowercase characters.

Required	No
Data type	Boolean

WebSphere MQ host:

WebSphere MQ server host defined by either its symbolic host name or an IP address.

Required	Yes
Data type	String

WebSphere MQ port:

This is the port that will be used when attempting to establish client transport mode connections to WebSphere MQ.

Required	Yes
Data type	Integer
Range	0 through 65535

Transport chain name:

The transport chain used to establish an outbound network connection to the WebSphere MQ server.

Required	Yes
Data type	Custom

WebSphere MQ channel:

The WebSphere MQ server connection channel, as defined in WebSphere MQ.

Required	Yes
Data type	String

Test connection:

Test the connection to WebSphere MQ.

In a single server environment, any attempt to test the connection fails if WebSphere MQ functionality has been disabled. In a network deployment environment, any attempt to test the connection fails if WebSphere MQ functionality has been disabled at either the cell or deployment manager node scope. For more information see “Disabling WebSphere MQ functionality in WebSphere Application Server” on page 868.

Messaging authentication alias:

WebSphere MQ server authentication alias name used when exchanging messages with WebSphere MQ.

Required	No
Data type	drop-down list

Trust user identifiers received in messages:

If selected user identifiers received as part of WebSphere MQ messages will be used for security purposes within the bus.

Select this option if you do not want the user IDs in messages to be overwritten with the administrative name of the WebSphere MQ server.

Required	No
Data type	Boolean

Automatic discovery of resources :

When selected this enables the discovery of WebSphere MQ queue names to assist the administrator.

Required	No
Data type	Boolean

Resource discovery authentication alias:

WebSphere MQ server authentication alias used for WebSphere MQ resource discovery.

Required	No
Data type	drop-down list

Reply-to queue :

WebSphere MQ queue name used by the system to obtain messages used in the resource discovery process.

Required	No
Data type	Text

Related Items

JAAS - J2C authentication data
Specifies a list of user identities and passwords for Java 2 connector security to use.

This item is not available until after the WebSphere MQ server definition has been created.

Service integration custom properties

Use custom properties to configure advanced settings for service integration objects such as messaging engines.

You can use the custom properties page to define the following service integration properties:

- “sib.msgstore.cachedDataBufferSize” on page 2258
- “sib.msgstore.discardableDataBufferSize” on page 2258
- “sib.msgstore.jdbcFailoverOnDBConnectionLoss” on page 2258
- “sib.msgstore.jdbcInitialDatasourceWaitTimeout” on page 2259
- “sib.msgstore.jdbcResAuthForConnections” on page 2259
- “sib.msgstore.jdbcStaleConnectionRetryDelay” on page 2259
- “sib.msgstore.storeFullWaitForCheckPoint” on page 2259
- “sib.msgstore.transactionSendLimit” on page 2260
-

- “sib.trm.retry” on page 2260
- “sib.wsrm.tokenLockTimeout” on page 2260

sib.msgstore.cachedDataBufferSize

The size in bytes of the data buffer that the messaging engine uses to contain data for which the quality of service attribute is *better than* best effort nonpersistent and that is held in the data store. The default value is 320000, which is approximately 320 kilobytes.

The purpose of the cached data buffer is to optimize the performance of the messaging engine by caching in memory the data that the messaging engine might otherwise have to read from the data store. As it writes data to the data store and reads from the data store, the messaging engine attempts to add that data to the cached data buffer. The messaging engine might discard data already in the buffer to make space.

Data type	Bytes
Default	40000000

sib.msgstore.discardableDataBufferSize

The size in bytes of the data buffer that the messaging engine uses to contain data for which the quality of service attribute is best effort nonpersistent. The default value is 320000, which is approximately 320 kilobytes.

The discardable data buffer contains all data for which the quality of service attribute is best effort nonpersistent. That data comprises both data that is involved in active transactions, and any other best effort nonpersistent data that the messaging engine has neither discarded nor consumed. The messaging engine holds this data entirely within this memory buffer and never writes the data to the data store. When the messaging engine adds data to the discardable data buffer, for example when the messaging engine receives a best effort nonpersistent message from a client, the messaging engine might discard data already in the buffer to make space. The messaging engine can discard only data that is not involved in active transactions. This behavior enables the messaging engine to discard best effort nonpersistent messages.

Increasing the size of the discardable data buffer allows more best effort nonpersistent data to be handled before the messaging engine begins to discard messages.

If the messaging engine attempts to add data to the discardable data buffer when insufficient space remains after discarding all the data that is not involved in active transactions, the messaging engine throws a `com.ibm.ws.sib.msgstore.OutOfCacheSpace` exception. Client applications can catch this exception, wrapped inside API-specific exceptions such as javax.jms.JMSException.

Data type	Bytes
Default	1280000

sib.msgstore.jdbcFailoverOnDBConnectionLoss

The property determines the behavior of the messaging engine and its hosting server in the event that the connection to the data store is lost.

Property value	**Behavior when the data store connection is lost**
true (default)	The server shuts down and must be manually restarted.

Property value	Behavior when the data store connection is lost
false	The messaging engine continues to run and accept work, and periodically attempts to regain the connection to the data store. If work continues to be submitted to the messaging engine while the data store is unavailable, the results can be unpredictable, and the messaging engine can be in an inconsistent state when the data store connection is restored. **Note:** If work continues to be submitted to the messaging engine, even nonpersistent messaging can fail because the messaging engine might need to use the data store, for example to allocate a unique ID to a message, or to move nonpersistent messages out of memory.

sib.msgstore.jdbcInitialDatasourceWaitTimeout

The time, in milliseconds, to wait for the data store to become available. This time includes the time required to establish a connection to the database and to obtain the required table locks.

Data type	Milliseconds
Default	900000 (15 minutes)

sib.msgstore.jdbcResAuthForConnections

The messaging engine resource authorization mechanism used when sharing connections. Default value is Container.

Data type	String
Default	Container

sib.msgstore.jdbcStaleConnectionRetryDelay

The time, in milliseconds, to wait between attempts to connect to the data store.

For example, if you set the sib.msgstore.jdbcInitialDatasourceWaitTimeout property to 600000, and the sib.msgstore.jdbcStaleConnectionRetryDelay property to 3000, the messaging engine will attempt to connect every 3 seconds for 10 minutes.

Data type	Milliseconds
Default	2000 (2 seconds)

sib.msgstore.storeFullWaitForCheckPoint

This property determines the action a messaging engine takes when a file store is full and applications try to send further messages.

When a file store is full, the messaging engine carries out a checkpoint of the log file to reconcile all message sends and receives since the last checkpoint. This process might take some time to complete. Between the time when the file store becomes full and the time when the checkpoint is complete, if applications try to send a message, the messaging engine throws the exception ObjectStoreFullException and issues message CWSOM1042E.

When an application thread that is sending a message finds that the file store is full, it requests a checkpoint. The default behavior, with the property value set to false, is that the application thread then throws the exception ObjectStoreFullException to the application immediately and returns. You can select an alternative behavior by setting the property value to true. With this property value, the application thread does not throw the exception, but waits until the checkpoint has completed. If the checkpoint frees space in the file store, the application thread proceeds and sends the messages before returning. If the file store is still full after the checkpoint, the application thread throws the exception to the application.

Set the property value to true, and make application threads wait for the checkpoint to be completed, if your applications delete all the messages in the file store, and so they logically know that the file store is no longer full. Although the applications must still wait until the checkpoint is complete, they do not receive exceptions while the checkpoint is being carried out, and they do not have to retry the send.

Data type	Boolean
Default	False

sib.msgstore.transactionSendLimit

The maximum number of operations that the messaging engine includes in each transaction. For example, each JMS send or receive is an operation that counts towards the transaction send limit. The default value is 100.

Data type	Integer
Default	100

sib.trm.retry

The messaging engine to messaging engine connection retry interval, in seconds. The retry interval is the time delay left between attempts to contact neighboring messaging engines with which communications exist. The default retry interval is 30 seconds.

Data type	Seconds
Default	30

sib.wsrm.tokenLockTimeout

This property affects WS-ReliableMessaging managed qualities of service. Set this property on the messaging engine that is specified in the policy binding for the WS-ReliableMessaging application.

This property is the amount of time, in milliseconds, that a lock is held on a WS-ReliableMessaging message. If a server fails while processing a message, the lock is released at the end of this timeout period, so that other servers can continue the processing. If the original server recovers before the timeout period ends, it continues processing the message. The lock is released at the end of the timeout period even if the message is still being processed.

If your system is processing large messages, you might want to increase the value of this property. For example, if a message takes 12 minutes to process, the lock is released 2 minutes before processing is complete. To avoid this situation, change the property to a value that is greater than 12 minutes.

If your system is processing small messages, you might want to decrease the value of this property, so that if a failure occurs the lock is released more quickly, and other servers can continue the processing without delay.

Data type	Milliseconds
Default	600000 (10 minutes)

SIBAdminCommands: Bus administrative commands for the AdminTask object

You can use these administrative commands to manage service integration buses.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

createSIBus command

Use the createSIBus command to create a new service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a new service integration bus.

Target object

None

Required parameters

-bus *busname*
The name by which you want the service integration bus to be known. Choose a unique name.

Conditional parameters

None.

Optional parameters

-description *text*
An optional description for the bus, for administrative purposes.

-secure TRUE | FALSE
This parameter is deprecated for this version.

interEngineAuthAlias *name*
The name of the authentication alias used to authorize communication between messaging engines on the bus.

You must specify an inter-engine authentication alias if the bus contains a WebSphere Application Server Version 6.x bus member. When bus security is enabled, the bus uses the inter-engine authentication alias to authenticate incoming connections from other messaging engines. An unauthorized messaging engine cannot connect to the bus.

-mediationsAuthAlias *name*
The name of the authentication alias used to authorize mediations to access the bus.

-securityGroupCacheTimeout *timeout_value*
The length of time, in minutes, that a security group will be cached for. *timeout_value* can be in the range 0 through 99999. The default value is 120.

Increasing the timeout decreases the load on the user registry and improves performance but makes the system less responsive to changes in a user's group membership. To tune the group cache to the optimum setting, you need to balance the need for responsiveness with the registry load. For example, if the system must respond quickly to changes in a user's group membership, specify a timeout of 15 minutes. If the system needs to respond less frequently, for example to respond to overnight changes, specify a timeout of 1440 minutes (24 hours). With a setting of 0, entries in the cache do not timeout, and so remain until the server is next restarted.

A change to this value is effective immediately and only affects the group cache of the bus for which the configuration was changed.

-protocol *protocol*
The transport chain used for communication between messaging engines in this bus.

The transport chain must correspond to one of the transport chains defined in the Messaging engine inbound transports settings for the server. All servers automatically have a number of transport chains defined to them, and it is also possible to create new transport chains.

The default transport chain is InboundBasicMessaging.

-discardOnDelete TRUE | FALSE
Set this option to TRUE if messages on a deleted message point can be discarded. Set this option to FALSE if messages on a deleted message point should be retained at a system exception destination.

highMessageThreshold *number*
The maximum total number of messages that any messaging engine on the bus can place on its message points.

This value is used to set the default high message threshold for a messaging engine when the messaging engine is created.

configurationReloadEnabled TRUE | FALSE
Set this option to TRUE to dynamically reload configuration files for this bus.

When this option is TRUE, certain changes to the bus configuration are applied without requiring the messaging engines to be restarted. These changes are applied when destinations or mediations are added to, or removed from, the bus. This option also controls whether automatic updates occur on all the messaging engines on the bus.

-busSecurity TRUE | FALSE
Set this option to TRUE to enforce the authorization policy for the bus, which also requires administrative security to be enabled. Set this option to FALSE if you always want to disable bus security. If administrative security is disabled, the bus is insecure.

-bootstrapPolicy SIBSERVICE_ENABLED | MEMBERS_AND_NOMINATED | MEMBERS_ONLY
Set one of three options to enforce a bus members policy for the bus.

SIBSERVICE_ENABLED
Any server in the cell that has the SIB service enabled can service bootstrap requests.

MEMBERS_AND_NOMINATED
Only bus members or a nominated bootstrap server can service bootstrap requests.

MEMBERS_ONLY
Only bus members can service bootstrap requests.

-useServerIdForMediations TRUE | FALSE
Set this option to TRUE if you want to run mediations using a single server identity for the bus. This option enables you to run mediations across multiple security domains without the need to specify a mediation authentication alias for each domain. You can use a server identity to run mediations on the global domain. Set this option to FALSE if you want to run mediations using a mediations authentication alias.

-auditAllowed TRUE | FALSE
Set this option to be TRUE to enable security auditing for the bus. Set this option to be FALSE to disable security auditing for the bus. The default value is TRUE. You must have Audit Administrator privileges to use this parameter.

Example

```
AdminTask.createSIBus('[-bus bus1 -description [A new bus] -busSecurity false ]')
'bus1(cells/cell01/buses/bus1|sib-bus.xml#SIBus_1213019988044)'
```

deleteSIBus command

Use the deleteSIBus command to delete a specified service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes a specified service integration bus, and deletes the artifacts that are related to the bus including queue points, publication points, and mediation points. This command also deletes the new artifacts that are created when queues are assigned or mediated to a WebSphere MQ server bus member, including the WebSphere MQ server bus members. This command does not delete the following:

- Queue managers that are associated with the bus through WebSphere MQ server definitions.
- Messages residing on WebSphere MQ queue managers.
- WebSphere MQ queues.

CAUTION:
This command deletes everything on and related to the bus. There is no confirmation prompt before deleting the bus.

Target object

The specified service integration bus.

Required parameters

-bus *busname*
The name by which the service integration bus is known. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.deleteSIBus('[-bus bus1 ]')
```

listSIBuses command

Use the listSIBuses command to list all service integration buses for a given scope.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists the names of all service integration buses for a given scope.

Target object

A given scope.

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listSIBuses()
'bus1(cells/cell01/buses/bus1|sib-bus.xml)
bus2(cells/cell01/buses/bus2|sib-bus.xml)'
```

modifySIBus command

Use the modifySIBus command to modify the properties of a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command modifies the properties of a service integration bus.

Target object

A bus.

Required parameters

-bus *busname*
The name by which the service integration bus is known. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

-description *text*
An optional description for the bus, for administrative purposes.

-secure TRUE | FALSE
This parameter is deprecated for this version.

-mediationsAuthAlias *name*
The name of the authentication alias used to authorize mediations to access the bus.

-securityGroupCacheTimeout *timeout_value*
The length of time, in minutes, that a security group will be cached for. *timeout_value* can be in the range 0 through 99999. The default value is 120.

Increasing the timeout decreases the load on the user registry and improves performance but makes the system less responsive to changes in a user's group membership. To tune the group cache to the optimum setting, you need to balance the need for responsiveness with the registry load. For example, if the system must respond quickly to changes in a user's group membership, specify a timeout of 15 minutes. If the system needs to respond less frequently, for example to respond to overnight changes, specify a timeout of 1440 minutes (24 hours). With a setting of 0, entries in the cache do not timeout, and so remain until the server is next restarted.

A change to this value is effective immediately and only affects the group cache of the bus for which the configuration was changed.

-protocol *protocol*
The transport chain used for communication between messaging engines in this bus.

-discardOnDelete TRUE | FALSE
Set this option to TRUE if messages on a deleted message point can be discarded. Set this option to FALSE if messages on a deleted message point should be retained at a system exception destination.

-busSecurity TRUE | FALSE
Set this option to TRUE to enforce the authorization policy for the bus, which also requires administrative security to be enabled. Set this option to FALSE if you always want to disable bus security. If administrative security is disabled the bus is not secured.

-permittedChains ALL | SSL_ENABLED | LISTED
Set one of three options to enforce a transport policy for the bus.

ALL Allow the use of all defined transport channel chains.

SSL_ENABLED
Restrict the use of defined transport channel chains to those protected by SSL.

LISTED
Restrict the use of defined transport channel chains to the list of protected transports.

-bootstrapPolicy SIBSERVICE_ENABLED | MEMBERS_AND_NOMINATED | MEMBERS_ONLY
Set one of three options to enforce a bus members policy for the bus.

SIBSERVICE_ENABLED
Any server in the cell that has the SIB service enabled can service bootstrap requests.

MEMBERS_AND_NOMINATED
Only bus members or a nominated bootstrap server can service bootstrap requests.

MEMBERS_ONLY
Only bus members can service bootstrap requests.

-useServerIdForMediations TRUE | FALSE
Set this option to TRUE if you want to run mediations using a single server identity for the bus. This option enables you to run mediations across multiple security domains without the need to specify a mediation authentication alias for each domain. You can use a server identity to run mediations on the global domain. Set this option to FALSE if you want to run mediations using a mediations authentication alias.

-auditAllowed TRUE | FALSE
Set this option to be TRUE to enable security auditing for the bus. Set this option to be FALSE to disable security auditing for the bus. The default value is TRUE. You must have Audit Administrator privileges to use this parameter.

Example

```
AdminTask.modifySIBus('[-bus bus1 -description [A new description of the bus]]')
'bus1(cells/cell01/buses/bus1|sib-bus.xml)'
```

showSIBus command

Use the showSIBus command to show the properties of a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

This command lists details about the properties of a service integration bus.

Target object

A bus.

Required parameters

-bus *busname*
The name by which the service integration bus is known. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.showSIBus('[-bus bus1]')
'{secure=false, useServerIdForMediations=false, discardOnDelete=false,
auditAllowed=true, highMessageThreshold=50000, busName=bus1,
securityGroupCacheTimeout=120, configurationReloadEnabled=true,
bootstrapPolicy=SIBSERVICE_ENABLED, permittedChains=ALL}'
```

addSIBusMember command

Use the addSIBusMember command to add a member to a service integration bus by using the wsadmin tool. A bus member can be an application server or a WebSphere MQ server.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The addSIBusMember command adds a new member to a service integration bus. When an application server is added as a member of a bus, a messaging engine with default settings is created automatically.

When a WebSphere MQ server is added as a member of a bus, a server proxy is created that can override the parent connection properties (host, port, channel, and SSL security configuration alias) that are defined in the WebSphere MQ server. By this means, a different set of connection properties can be assigned to each bus membership.

Target object

None.

A bus member object is created. If the bus member is a server, a messaging engine is also created.

Required parameters

-bus *bus_name*
The name by which the service integration bus is known. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

-node *node_name*

-server *server_name*

To add an application server as a bus member, specify both the name of the node on which the server runs, and the name of the server.

If you specify these parameters, do not specify the **-wmqServer** parameter.

-wmqServer

To add a WebSphere MQ server as a bus member, specify the name of the WebSphere MQ server. This is the name that was specified in the **-name** parameter when the WebSphere MQ server was created.

If you specify this parameter, do not specify the **-node** or **-server** parameters.

Optional parameters

-description *text*

An optional description for the bus member, for administrative purposes.

-virtualQueueManagerName

The override value for the name of the WebSphere MQ server virtual queue manager. When sending messages to WebSphere MQ, the WebSphere MQ gateway queue manager sees the bus as a remote queue manager. The virtual queue manager name is the name that is passed to WebSphere MQ as the name of this remote queue manager. The default value is the name of the bus. If this value is not a valid name for a WebSphere MQ queue manager, or if another WebSphere MQ queue manager already exists that has the same name, then replace the default value with another value that is a valid and unique name for a WebSphere MQ queue manager. To be valid, the name must meet the following criteria:

- It must contain between 1 and 48 characters.
- It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

-host

The override value for the WebSphere MQ server bus member host attribute. This value is the host name or IP address of the host to which a connection is established to communicate with a queue manager or queue-sharing group. Use this parameter if the **-wmqServer** parameter is specified.

-port

The override value for the WebSphere MQ server bus member port attribute. The port number is monitored by a queue manager listener or queue-sharing group listener, which is listening for connections. The default value is 1414. Use this parameter if the **-wmqServer** parameter is specified.

-channel

The override value for the WebSphere MQ server bus member channel attribute. This is the name of the server connection channel that is used to establish a connection to the WebSphere MQ queue manager or WebSphere MQ for z/OS queue sharing group. The default value is SYSTEM.DEF.SVRCONN. Use this parameter if the **-wmqServer** parameter is specified.

-securityAuthAlias

The override value for the WebSphere MQ server bus member **securityAuthAlias** attribute. This is the authentication alias that is supplied when connecting to the WebSphere MQ server. This parameter has no default. Use this parameter if the **-wmqServer** parameter is specified.

-transportChain

The override value for the WebSphere MQ server bus member **transportChain** attribute. This is the name of the transport chain to use when communicating with WebSphere MQ. The default value is OutboundBasicWMQClient. Use this parameter if the **-wmqServer** parameter is specified.

-trustUserIds TRUE | FALSE
The override value for the WebSphere MQ server bus member **trustUserIds** attribute. This determines whether user IDs that are received in messages from WebSphere MQ are propagated into messages. The application user ID is always set from the **jsAppUserIdRFH2** value. If this is not present (either because the key/value pair is not present in the <sib> folder of the RFH2 header, or because the message does not have a RFH2 header), this field is not set. This parameter has two possible values:

TRUE User IDs are propagated into messages.

FALSE
User IDs are not propagated into messages.

The default value is TRUE. Use this parameter if the **-wmqServer** parameter is specified.

-fileStore
Create a file store to use as a message store for the messaging engine. A file store is a type of message store that directly uses files in a file system through the operating system. The alternative is to use a data store. For more information, see the related links.

-logSize *logsize*
The size of the log file in MB. Use this parameter if the **-fileStore** parameter is specified.

-logDirectory *logdirectoryname*
The name of the log file directory if you do not want to use the default log directory. Use this parameter if the **-fileStore** parameter is specified.

-minPermanentStoreSize *minpermanentstoresize*
The minimum size of the permanent store file in MB. Use this parameter if the **-fileStore** parameter is specified.

-minTemporaryStoreSize *mintemporarystoresize*
The minimum size of the temporary store file in MB. Use this parameter if the **-fileStore** parameter is specified.

-maxPermanentStoreSize *maxpermanentstoresize*
The maximum size of the permanent store file in MB. Use this parameter if the **-fileStore** parameter is specified.

-maxTemporaryStoreSize *maxtemporarystoresize*
The maximum size of the temporary store file in MB. Use this parameter if the **-fileStore** parameter is specified.

-unlimitedPermanentStoreSize TRUE | FALSE
A parameter that specifies whether the permanent store size is unlimited. This parameter has two possible values:

TRUE The permanent store size is unlimited.

FALSE
The permanent store size is limited. If you use this option, supply a **-maxPermanentStoreSize** parameter.

Use this parameter if the **-fileStore** parameter is specified.

-unlimitedTemporaryStoreSize TRUE | FALSE
A parameter that specifies whether the temporary store size is unlimited. This parameter has two possible values:

TRUE The temporary store size is unlimited.

FALSE
The temporary store size is limited. If you use this option, supply a **-maxTemporaryStoreSize** parameter.

Use this parameter if the **-fileStore** parameter is specified.

-permanentStoreDirectory *permanentstoredirectoryname*
The name of the permanent store directory if you do not want to use the default permanent store directory. Use this parameter if the **-fileStore** parameter is specified.

-temporaryStoreDirectory *temporarystoredirectoryname*
The name of the temporary store directory if you do not want to use the default temporary store directory. Use this parameter if the **-fileStore** parameter is specified.

-dataStore
Create a data store to use as a message store for the messaging engine.

A data store consists of the set of tables that a messaging engine uses to store persistent data in a database. See "Data store tables" on page 1967 for a list of the tables that comprise a data store. All the tables in a data store are held in the same database schema. You can create multiple data stores in the same database, provided that you use a different schema name for each data store. The alternative is to use file store (the default). For more information, see the related links.

-createDefaultDatasource TRUE | FALSE
A parameter that specifies whether to create a default data source when the messaging engine is created. This parameter has two possible values:

TRUE Create a default data source.

FALSE
Do not create a default data source.

Use this parameter if the **-dataStore** parameter is specified. Do not use this parameter if the **-cluster** parameter is specified.

-datasourceJndiName *jndiname*
The JNDI name of the data source that the messaging engine uses to access the relational database management system (RDBMS) for the data store. Use this parameter if the **-dataStore** parameter is specified.

-authAlias *authalias*
The name of the authentication alias that the messaging engine uses to connect to the database in its data store. Use this parameter if the **-dataStore** parameter is specified.

-createTables *datasource*
Create the database tables for the specified data source automatically. If this option is not specified, the database administrator must create the tables. Use this parameter if the **-dataStore** parameter is specified.

-schemaName *schemaname*
The name of the database schema that contains the tables for the data store, if you do not want to use the default schema name. For details about the default schema, see "Creating users and schemas in the database" on page 1959. Use this parameter if the **-dataStore** parameter is specified.

-initialHeapSize *size*
The initial Java virtual machine (JVM) heap size, in megabytes, of the server or of each server in a cluster. There is no default value. If this parameter is not specified, no change is made. If this parameter is specified, the supplied value must be greater than or equal to zero and less than or equal to 2048. If the parameter is supplied without a value, an error message is generated.

-maxHeapSize *size*
The maximum JVM heap size, in megabytes, of the server or of each server in a cluster. There is no default value. If this parameter is not specified, no change is made. If this parameter is specified, the supplied value must be greater than or equal to zero and less than or equal to 2048. If the parameter is supplied without a value, an error message is generated.

Examples

Add a server1 on node1 as a member of bus1 with a default file store.

```
AdminTask.addSIBusMember('[-bus bus1 -node node1 -server server1 ]')
```

Add server1 as a member of bus1, and use a file store to save messages.

```
AdminTask.addSIBusMember('[-bus bus1 -node node1 -server server1 -fileStore ]')
```

Add server1 as a member of bus1, and use a file store to save messages, with options.

```
AdminTask.addSIBusMember('[-bus bus1 -node node1 -server server1 -fileStore
-logSize 100 -logDirectory C:\filestore1 ]')
```

Add server1 as a member of bus1, and use a data store to save messages.

```
AdminTask.addSIBusMember('[-bus bus1 -node node1 -server server1 -dataStore ]')
```

Add server1 as a member of bus1, and use a data store to save messages, with options.

```
AdminTask.addSIBusMember('[-bus bus1 -node node1 -server server1 -dataStore
-createDefaultDatasource true -datasourceJndiName myjndi]')
```

Change the initial JVM heap size to 256 and the maximum JVM heap size to 512.

```
AdminTask.addSIBusMember('[-bus bus1 -node node1 -server server1
-initialHeapSize 256 -maxHeapSize 512]')
```

listSIBusMembers command

Use the listSIBusMembers command to list the members of a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists the names of all members in a service integration bus. The list includes all WebSphere MQ servers that are bus members.

Target object

A service integration bus.

Required parameters

-bus *busname*

The name by which the service integration bus is known. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.listSIBusMembers('[-bus Bus1 ]')

[cells/9994GKCCell/buses/Bus1|sib-bus.xml#SIBusMember_1092155259869]
[cells/9994GKCCell/buses/Bus1|sib-bus.xml#SIBusMember_1092159844593]
[cells/9994GKCCell/buses/Bus1|sib-bus.xml#SIBusMember_1092160253751]
```

---

removeSIBusMember command

Use the removeSIBusMember command to remove a member from a service integration bus by using the wsadmin tool. A bus member can be an application server or a WebSphere MQ server.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The removeSIBusMember command removes a bus member from the named bus. It cleans up artifacts that are associated with the removed membership, for example, associated WebSphere MQ server bus member artifacts, mediation execution points, or core group policies for messaging engines that are associated with the bus member. To remove a bus member by using this command, specify one of the following resources:

- The server and the node on which that server runs
- The WebSphere MQ server

This command does not delete messages from the associated WebSphere MQ queues, and it does not delete the queues.

Target object

A service integration bus.

Required parameters

-bus *bus_name*
The name by which the service integration bus is known. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

-node *node_name*
-server *server_name*
To remove an application server from a bus, specify both the name of the node on which the server runs and the name of the server.

If you specify these parameters, do not specify the **-wmqServer** parameter.

-wmqServer *mqservername*
To remove a WebSphere MQ server from a bus, specify the name of the WebSphere MQ server. This is the name that was specified in the **-name** parameter when the WebSphere MQ server was created.

If you specify this parameter, do not specify the **-node** or **-server** parameters.

Optional parameters

None.

Example

Remove server1 from bus1.

```
AdminTask.removeSIBusMember('[-bus bus1 -node node1 -server server1 ]')
```

showSIBusMember command

Use the showSIBusMember command to show the properties of a service integration bus member.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists the details of a specified member of a service integration bus. You can also use this command to list the details of a WebSphere MQ server bus member.

Target object

A bus member.

Required parameters

-bus *busname*
The name by which the service integration bus is known. You can use the listSIBuses command to list the names of existing buses.

-wmqServer
The name of the WebSphere MQ server bus member.

Conditional parameters

-node *nodename* **-server** *servername*
To list details of an application server as a bus member, specify both the name of the node on which the server runs and the name of the server.

-cluster *cluster*
To list details of a server cluster, specify the name of the cluster.

Use this option only in WebSphere Application Server environments that support server clusters.

Optional parameters

None.

Example

```
AdminTask.showSIBusMember(['-bus bus1 -node node1 -server server1'])
'{node=node1, server=server1}'
```

listAllSIBBootstrapMembers command

Use the listAllSIBBootstrapMembers command to list all the bootstrap members that can bootstrap into a selected bus.

You can use this command to list all the bootstrap members for a specified bus, or you can use the administrative console, see “Listing the bootstrap members for a bus” on page 1895.

This command requires the name of the bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The listAllSIBBootstrapMembers command lists all the servers and clusters that can bootstrap into a selected bus.

Target object

Bootstrap members for a selected bus.

Required parameters

-bus
The name of the bus for which you want to list all the bootstrap members. The data type is string. There is no default value.

Conditional parameters

None

Optional parameters

None

Example

The following example lists all the bootstrap members of `bus1`:

```
AdminTask.listAllSIBBootstrapMembers('[-bus bus1]')
```

listSIBNominatedBootstrapMembers command

Use the listSIBNominatedBootstrapMembers command to list all the nominated bootstrap members for a specified bus.

You can use this command to list all the nominated bootstrap members for a specified bus, or you can use the administrative console, see “Listing the bootstrap members for a bus” on page 1895.

This command requires the name of the bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The listSIBNominatedBootstrapMembers command lists all the nominated bootstrap members for a specified bus.

Target object

Nominated bootstrap members.

Required parameters

-bus
The name of the bus for which you want to list all nominated bootstrap members. The data type is string. There is no default value.

Conditional parameters

None

Optional parameters

None

Example

The following example lists the nominated bootstrap members for `bus1`:

```
AdminTask.listSIBBootstrapMembers('[-bus bus1]')
```

addSIBBootstrapMember command

Use the addSIBBootstrapMember command to add a nominated bootstrap member to a specified bus.

You can use this command to add a nominated bootstrap member to a specified bus, or you can use the administrative console: “Nominating bootstrap members for a bus” on page 1896.

This command requires the name of a bus, and a node and server name, or a cluster name.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The addSIBBootstrapMember command adds a nominated bootstrap member to the specified bus.

Target object

A nominated bootstrap member.

Required parameters

-bus
The name of the bus to which the nominated bootstrap member is added.

Conditional parameters

None

Optional parameters

-node
The name of the node on which the server you want to add exists. The data type is string. You must specify either the **node** or the **cluster** parameter. If you specify the **node** parameter, you must also specify the **server** parameter.

-server
The name of the server that you want to add as a nominated bootstrap member. The data type is string. You must only use this parameter with the **node** parameter, not with the **cluster** parameter.

-cluster
The name of the cluster that you want to add as a nominated bootstrap member. The data type is string. You must specify either the **cluster** or the **node** parameter. You cannot use the **cluster** parameter with the **node** or the **server** parameter.

Example

The following example adds `server1` on `node1` as a nominated bootstrap member of `bus1`:

```
AdminTask.addSIBBootstrapMember('[-bus bus1 -node node1 -server server 1]')
```

removeSIBBootstrapMember command

Use the removeSIBBootstrapMember command to remove a nominated bootstrap member from a selected bus.

You can use this command to remove a nominated bootstrap member from a selected bus, or you can use the administrative console: “Deleting nominated bootstrap members from a bus” on page 1896.

This command requires the name of a bus, and either the node and server name, or the cluster name, of an existing bootstrap member.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The removeSIBBootstrapMember command removes a nominated bootstrap member from a selected bus.

Target object

The nominated bootstrap member is removed from the selected bus.

Required parameters

-bus
The name of the bus from which you want to remove the nominated bootstrap member. The data type is string.

Conditional parameters

None

Optional parameters

-node
The name of the node where the server you want to remove exists. The data type is string. You must specify either the **node** or the **cluster** parameter. If you specify the **node** parameter, you must also specify the **server** parameter.

-server
The name of the server that you want to remove. The data type is string. You must only use this parameter with the **node** parameter, not with the **cluster** parameter.

-cluster
The name of the cluster that you want to remove. The data type is string. You must specify either the **cluster** or the **node** parameter. You cannot use the **cluster** parameter with the **node** or the **server** parameter.

Examples

The following example removes a nominated bootstrap member `cluster1` from `bus1`:

```
AdminTask.removeSIBBootstrapMember('[-bus bus1 -cluster cluster1]')
```

SIBAdminCommands: Foreign bus administrative commands for the AdminTask object

You can use these administrative commands to manage foreign buses. A foreign bus represents a service integration bus with which another service integration bus can exchange messages.

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and Jython.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

createSIBForeignBus command

Use the createSIBForeignBus command to create a new service integration foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a new foreign bus for a specified service integration bus. A foreign bus represents a bus with which another bus can exchange messages.

Target object

A specified service integration bus.

Required parameters

-bus *busname*
The name of the service integration bus for which you want to create the foreign bus. You can use the listSIBuses command to list the names of existing buses.

-name *busname*
The name by which you want the foreign bus to be known.

Notes:

- When you create a foreign bus that represents another service integration bus, the name of the foreign bus must match the name of the other service integration bus.
- When you intend to link two buses directly, you must assign them unique names.
- You must not change the name of a foreign bus name after it has been configured.

-routingType Direct | Indirect
Create a foreign bus with the physical link (also known as the *routing type*) specified:

Direct A *service integration bus link* from a messaging engine in the local bus to a messaging engine in the foreign bus.

Indirect
An *indirect link*, that is, a link that is made through one or more intermediate foreign buses.

Conditional parameters

None.

Optional parameters

-description *text*
An optional description for the bus, for administrative purposes.

-type MQ | SIBus
Create a foreign bus with the type specified:

MQ Create a foreign bus to link the service integration bus to a WebSphere MQ network.

SIBus Create a foreign bus to link the service integration bus to another service integration bus.

-sendAllowed True | False
(Default: True) Whether producers can send messages to the foreign bus.

False Producers cannot send messages to the foreign bus.

True Producers can send messages to the foreign bus.

-inboundUserid *userID*
The inbound user ID is used to authorize inbound messages sent from a foreign bus to destinations in a secure service integration bus. If the bus is not secure, the inbound user ID property has no affect on messages. You might want to specify an inbound user ID for use in the following scenarios:

- The foreign bus and the secure service integration bus are in different security domains, and the foreign bus user IDs are not recognized by the secure bus.
- You want local control over access to the secure bus by inbound messages.

Note that if the receiving service integration bus is secure but the foreign bus is not secure, and an inbound user ID is not set, an inbound message from the foreign bus is only authorized to destinations that allow unauthenticated users access.

-outboundUserid *userID*
The outbound user ID replaces the user ID that identifies the source of a message in all messages

being sent to the foreign bus. This user ID is also used by the foreign bus to authorize the message to its destination if both buses are secure buses and the foreign bus has not overridden the user ID with its own inbound user ID.

-nextHopBus *bus_name*
(If **-routingType** is Indirect) The name of the next service integration bus in the sequence of connected buses. An intermediate bus can be a WebSphere MQ system rather than a service integration bus.

-topicSpaceMappings *local topicSpace_name : remote topicSpace_name*
A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. For publications to flow from the local topic space into the foreign bus, an equivalent topic space mapping is required by the foreign bus. Topic space names for the local bus are mapped to topic space names defined on the foreign bus. It is common for these two names to match. Note that mapping two topic spaces implies that the topics within them are the same. You can specify multiple pairs of topic spaces.

Example

```
AdminTask.createSIBForeignBus('[-bus bus1 -name foreignbus1 -routingType Direct
-type SIBus]')
'foreignbus1(cells/cell01/buses/bus1|sib-bus.xml#SIBForeignBus_1213023645293)'
```

deleteSIBForeignBus command

Use the deleteSIBForeignBus command to delete a foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes a foreign bus. The command does not affect the service integration bus for which the foreign bus was created.

CAUTION:
This command deletes the foreign bus and its additional property configuration. Although there is no confirmation prompt before deleting the bus, the bus is not deleted from the master configuration until you save your changes.

Target object

A foreign bus.

Required parameters

-bus *busname*
The name of the service integration bus for which you want to delete a foreign bus. You can use the listSIBuses command to list the names of existing buses.

-name *busname*
The name of the foreign bus that you want to delete. You can use the listSIBForeignBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.deleteSIBForeignBus('[-bus bus1 -name foreignbus1]')
```

listSIBForeignBuses command

Use the listSIBForeignBuses command to list all foreign buses for a specified service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists the names of all foreign buses for a specified service integration bus. A foreign bus represents a bus in another cell (or within the same cell) or a WebSphere MQ network, with which a service integration bus can exchange messages.

Target object

A bus.

Required parameters

-bus *busname*
The name of the service integration bus for which you want to list foreign buses.

Conditional parameters

None.

Optional parameters

-routingType Direct | Indirect
List the foreign buses with the physical link (also known as the *routing type*) specified:

Direct A *service integration bus link* from a messaging engine in the local bus to a messaging engine in the foreign bus.

Indirect
An *indirect link*, that is, a link that is made through one or more intermediate foreign buses.

-type MQ | SIBus
List the foreign buses with the type specified:

MQ List the foreign buses that link the service integration bus to a WebSphere MQ network.

SIBus List the foreign buses that link the service integration bus to another service integration bus.

Example

```
AdminTask.listSIBForeignBuses('[-bus Bus1]')
```

modifySIBForeignBus command

Use the modifySIBForeignBus command to modify the properties of a foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command modifies the properties of a foreign bus. A foreign bus represents a bus in another cell (or within the same cell) or a WebSphere MQ network, with which a service integration bus can exchange messages.

Target object

A foreign bus.

Required parameters

-bus *busname*
The name of the service integration bus for which you want to modify a foreign bus. You can use the listSIBuses command to list the names of existing buses.

-name *busname*
The name of the foreign bus that you want to modify. You can use the listSIBForeignBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

-description *text*
An optional description for the bus, for administrative purposes.

-sendAllowed False | False
(Default: True) Whether producers can send messages to the foreign bus.

False Producers cannot send messages to the foreign bus.

True Producers can send messages to the foreign bus.

-inboundUserid *userID*
The inbound user ID is used to authorize individual messages arriving from the foreign bus to destinations in the service integration bus. If this is not a secure bus, this property has no affect on messages. You might want to specify an inbound user ID:

- if the foreign bus is in a different security domain from this bus and user IDs from the foreign bus are not recognized in this bus
- to locally-control access of inbound messages to this bus.

If this is a secure bus and the foreign bus is not secure, and no inbound user ID is set, any inbound messages from the foreign bus are only authorized to destinations that allow unauthenticated users access.

-outboundUserid *userID*
The outbound user ID replaces the user ID that identifies the source of a message in all messages being sent to the foreign bus. This user ID is also used by the foreign bus to authorize the message to its destination if both buses are secure buses and the foreign bus has not overridden the user ID with its own inbound user ID.

-nextHopBus *bus_name*
(If -**routingType** is Indirect) The name of the next service integration bus in the sequence of connected buses. An intermediate bus can be a WebSphere MQ system rather than a service integration bus.

-topicSpaceMappings *local topicSpace_name : remote topicSpace_name*
A topic space mapping allows subscribers on the local topic space to receive messages published in the foreign topic space. For publications to flow from the local topic space into the foreign bus, an equivalent topic space mapping is required by the foreign bus. Topic space names for the local bus are mapped to topic space names defined on the foreign bus. It is common for these two names to match. Note that mapping two topic spaces implies that the topics within them are the same. You can specify multiple pairs of topic spaces.

Example

The example below specifies the following additional optional properties for an existing foreign bus called *foreignbus1*:

- An inbound user identity called *iuserid* to authorize messages sent from *foreignbus1* to the local bus, when security is enabled on *foreignbus1*, and the local bus.
- An outbound user identity called *ouserid* to authorize messages sent from the local bus to *foreignbus1*, when security is enabled on *foreignbus1*, and the local bus.

- A topic space mapping between *Topic.Space1* on the local bus and *Topic.Space2* on *foreignbus1*. The mapping allows subscribers on *Topic.Space1* on the local bus to receive messages published in *Topic.Space2* on *foreignbus1*.

```
AdminTask.modifySIBForeignBus('[-bus bus1 -name foreignbus1 -inboundUserid iuserid
-outboundUserid ouserid -topicSpaceMappings [[Topic.Space1 Topic.Space2]]]')
'foreignbus1(cells/cell01/buses/bus1|sib-bus.xml#SIBForeignBus_1213023645293)'
```

showSIBForeignBus command

Use the showSIBForeignBus command to show the properties of a foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists details about the properties of a foreign bus. A foreign bus represents a bus in another cell (or within the same cell) or a WebSphere MQ network, with which a service integration bus can exchange messages.

Target object

A foreign bus.

Required parameters

-bus *busname*
The name of the service integration bus for which you want to show a foreign bus. You can use the listSIBuses command to list the names of existing buses.

-name *busname*
The name of the foreign bus that you want to show. You can use the listSIBForeignBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.showSIBForeignBus('[-bus bus1 -name foreignbus1]')
'Foreign Bus Name = foreignbus1
Foreign Bus Uuid = 17B07B718182CDF6A73E75D2
Foreign Bus Description = A new foreign bus
Foreign Bus Send Allowed = true
mqRfh2 Allowed = false'
```

SIBAdminCommands: WebSphere MQ link administrative commands for the AdminTask object

You can use these administrative commands to manage WebSphere MQ links. A WebSphere MQ link connects a messaging engine as a queue manager to WebSphere MQ, thereby providing a bridge between a service integration bus and a WebSphere MQ network.

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and Jython.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

createSIBMQLink command

Use the createSIBMQLink command to create a new WebSphere MQ link for a specified service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The createSIBMQLink command creates a new WebSphere MQ link for a specified service integration bus. The WebSphere MQ link connects a messaging engine as a queue manager to WebSphere MQ, thereby providing a bridge between a service integration bus and a WebSphere MQ network.

Target object

A bus.

Required parameters

-bus
The name of the service integration bus for which you want to create the WebSphere MQ link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine for which you want to create the WebSphere MQ link. The WebSphere MQ link connects a messaging engine as a queue manager to WebSphere MQ, thereby providing a bridge between a service integration bus and a WebSphere MQ network.

-name
The name by which you want the WebSphere MQ link to be known.

-foreignBusName
The name of the foreign bus that defines the WebSphere MQ network for the WebSphere MQ link. You can use the listSIBForeignBuses command to list the names of existing foreign buses.

-queueManagerName
The name of the virtual queue manager associated with the messaging engine, and by which the messaging engine is known to a remote WebSphere MQ network.

The queue manager name must conform to the WebSphere MQ naming conventions; for example, the name must be a maximum of 48 characters.

-senderChannelTransportChain OutboundBasicMQLink | OutboundSecureMQLink
The name of the sender channel that sends messages to the gateway queue manager. The sender channel communicates with a WebSphere MQ receiver channel on the gateway queue manager, and converts service integration bus messages to MQ format messages.

Conditional parameters

None.

Optional parameters

WebSphere MQ link configuration parameters:

-description
An optional description for the bus, for administrative purposes.

-exceptionDestination
The destination for an inbound message when the WebSphere MQ link cannot deliver the message to its target bus destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist.

System
The WebSphere MQ link uses the default exception destination. All messages that cannot be delivered to the bus destination are rerouted automatically to the system default exception destination for the messaging engine that this link is assigned to: _SYSTEM.Exception.Destination*messaging_engine_name*.

None The WebSphere MQ link has no exception destination. Undeliverable messages are not rerouted to an exception destination and can block the processing of other messages waiting for delivery through the link to the same bus. This option can be used to preserve message ordering.

Specify
The WebSphere MQ link uses the exception destination specified here. If this is not possible, it uses the system exception destination.

-batchSize
The maximum number of messages that can be sent through a channel before taking a checkpoint.

The batch size does not affect the way the sender and receiver channels for this link transfer messages. Messages are always transferred individually, but are committed or backed out as a batch.

For more information about choosing the batch size, see the description of the batch size (**BATCHSZ**) property in the *Intercommunication* section of the WebSphere MQ information center.

Default 50
Range 1 through 9999

-maxMsgSize
The maximum message length, in bytes, that can be transmitted on any channel for the WebSphere MQ link. This is compared with the value for the corresponding partner WebSphere MQ channel and the actual maximum used is the lower of the two values.

For information about how to choose an appropriate number, see the description of the Maximum message length (**MAXMSGL**) property in the *Intercommunication* section of the WebSphere MQ information center.

Default 4194304 bytes (4MB)
Range 0 through 104857600

Specify 0 to use the largest value that the target queue manager will honor.

-heartBeat
The negotiated time, in seconds, between heartbeat flows passed from the WebSphere MQ link sender channel to the WebSphere MQ receiver channel when there are no messages on the transmission point being served by the WebSphere MQ link sender channel.

Heartbeats give the receiving channel the opportunity to quiesce the channel connection.

For more information about choosing the value for this property, see the *Intercommunication* section of the WebSphere MQ information center.

Default 300 seconds
Range 0 through 999999

-sequenceWrap
The value at which message sequence numbers wrap to start again at 1. For example, if you specify a value of 1000, when the message sequence number reaches 1001 it will restart at 1.

For more information about choosing the value for this property, see the description of the Sequence Number wrap (**SEQWRAP**) property in the *Intercommunication* section of the WebSphere MQ information center.

Default 999999999
Range 1 through 999999999

-nonPersistentMessageSpeed Fast | Normal
The class of service for nonpersistent messages on channels of this WebSphere MQ link.

Default Fast

Range

Fast Nonpersistent messages can be lost if there is a transmission failure or if the channel stops when the messages are in transit.

Normal
Nonpersistent messages are not lost if there is a transmission failure or if the channel stops when the messages are in transit.

-adoptable True | False
A property of the WebSphere MQ link, which shows whether a running instance of a WebSphere MQ link receiver channel (associated with this WebSphere MQ link) should be adopted. In the event of a communications failure, a running instance of a WebSphere MQ link receiver channel might be left waiting for messages. When communication is reestablished, and the partner WebSphere MQ sender channel next attempts to establish a session with the WebSphere MQ link receiver channel, the request will fail as there is already a running instance of the WebSphere MQ link receiver channel that believes it is in session with the partner WebSphere MQ sender channel. You can overcome this problem by selecting this option, which causes the already running instance of the WebSphere MQ link receiver channel to be stopped and a new instance to be started.

If you set this option to True, the WebSphere MQ sender channels might reestablish a connection to this WebSphere MQ link in the event that a communications failure has occurred and the link has not yet detected the failure condition.

Default True

-initialState Started | Stopped
The state of the WebSphere MQ link, which shows whether the link is started or stopped when the hosting messaging engine is first started. Until it is started, the WebSphere MQ link is unavailable.

Default Started

Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

WebSphere MQ link sender channel parameters:

-senderChannelName
The sender channel that sends messages to the gateway queue manager. The sender channel communicates with a WebSphere MQ receiver channel on the gateway queue manager, and converts service integration bus messages to MQ format messages.

This name must be the same as the name of the receiver channel on WebSphere MQ.

For more information about choosing channel names, see the description of the channel name (**CHANNEL**) property in the *Intercommunication* section of the WebSphere MQ information center.

-hostName
The host name or TCP/IP address for the gateway queue manager that is used to connect into the WebSphere MQ network.

Type the host name or IP address of the host on which the gateway queue manager runs.

If this field is blank, the gateway queue manager is assumed to be running on the same host as the messaging engine on which the WebSphere MQ link is defined.

-port
The TCP/IP port number on which the gateway queue manager is listening for the WebSphere MQ link.

Default	1414
Range	0 through 65535

-connameList
The connection name list for the gateway queue manager which is used to connect to the WebSphere MQ network. The connections are tried in the order in which they are specified in the connection name list until a connection is successfully established. If no connection is successful, the channel starts retry processing.

The connection names must be given as a comma separated list in the following format:

```
Dnsname1(portnumber1) , Dnsname2(portnumber2)
```

Type the DNS name or the IP address of the host on which the gateway queue manager is running. Type the port number in the range 0 through 65535.

If you do not specify the connection name list parameter, the gateway queue manager is assumed to be running on the same host as the messaging engine on which the WebSphere MQ link is defined.

If you do specify the connection name list parameter, you must have already specified a value for the sender channel name parameter, otherwise the value that you specify for the connection name list parameter is ignored.

If you specify the host name and port parameter as well as the connection name list parameter, then at runtime the connection name list takes precedence and host name and port values are ignored.

-discInterval
The time in seconds for which the sender channel waits for new messages to arrive on the transmission queue after sending a batch of messages. The channel disconnects after this interval, and must be restarted manually or by triggering.

The default value is a reasonable interval. Change this value only if you understand the implications for performance, and you need a different value for the requirements of the traffic flowing down your channels.

Performance is affected by the value specified for the disconnect interval. A very low value (a few seconds) can cause an unacceptable amount of processing in constantly starting up the channel. A very large value (more than an hour) might mean that system resources are unnecessarily held up.

If you want your channels to be active only when there are messages for them to transmit, you should set the disconnect interval to a fairly low value. Note that the default setting is quite high and so is not recommended for channels where this level of control is required. Because it is difficult to interrupt the receiving channel, the most economical option is to have the channel automatically disconnect and reconnect as the workload demands. For most channels, the appropriate setting of the disconnect interval can be established heuristically.

Default	900 seconds
Range	0 through 999999 A value of 0 (zero) means never disconnect; the channel waits indefinitely for messages.

-shortRetryCount
The maximum number of times that the sender channel tries to restart after a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then the long retry mechanism is invoked.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Default 10
Range 0 through 999999999

-shortRetryInterval
The number of seconds between attempts by the sender channel to restart after a communication or partner failure.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Default 60 seconds
Range 0 through 999999

-longRetryCount
The maximum number of times that the sender channel tries to restart after the short retry mechanism did not recover from a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then an error is logged and the channel is stopped.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Default 999999999
Range 0 through 999999999

-longRetryInterval
The number of seconds between attempts by the sender channel to restart after the short retry mechanism did not recover from a communication or partner failure.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Default 1200 seconds
Range 0 through 999999

-senderChannelInitialState Started | Stopped
The state of the WebSphere MQ link, which shows whether the sender channel is started or stopped when the associated WebSphere MQ link is first started. Until it is started, the channel is unavailable.

Default Started

Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

WebSphere MQ link receiver channel parameters:

-receiverChannelName
The name of the receiver channel for the WebSphere MQ link, used to receive messages from WebSphere MQ onto the bus.

This name must be the same as the name of the sender channel on WebSphere MQ.

-inboundNonPersistentReliability Best effort | Express | Reliable
The acceptable reliability of message delivery for nonpersistent message flows from WebSphere MQ through this WebSphere MQ link, from Best effort to Reliable, in order of increasing reliability.

This reliability delivery option is assigned to all WebSphere MQ nonpersistent messages flowing over this receiver channel.

Default
Reliable

Range

Best effort
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable
Messages are discarded when a messaging engine stops or fails.

-inboundPersistentReliability Reliable | Assured
The acceptable reliability of message delivery for inbound persistent message flows from WebSphere MQ through this WebSphere MQ link, from Reliable to Assured, in order of increasing reliability.

Default
Assured

Range

Reliable
Messages might be discarded when a messaging engine fails.

Assured
Messages are not discarded.

-receiverChannelInitialState Started | Stopped

The state of the WebSphere MQ link, which shows whether the receiver channel is started or stopped when the associated WebSphere MQ link is first started. Until it is started, the channel is unavailable.

Default Started

Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

-preferLocal TRUE | FALSE

This option indicates that the link prefers to send incoming messages to the queue point of the target destination that is located on the same messaging engine as the link, if available. The link must be owned by a messaging engine running on a WebSphere Application Server Version 7.0 or later server.

This option is supported only when used by a JMS application that is running with a WebSphere Application Server Version 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application Server Version 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Default TRUE

Range

TRUE Send each incoming message to the queue point of the target destination that is located on the same messaging engine as the link, if available.

FALSE Send incoming messages to any queue points of the target destinations.

Examples

```
AdminTask.createSIBMQLink('[-bus bus1 -messagingEngine cluster1.000-bus1
-name myMQLink2 -foreignBusName MQNetwork2 -queueManagerName MQMgrIPL
-senderChannelTransportChain OutboundBasicMQLink]')
'myMQLink2(cells/cell01/nodes/node01/servers/server1|sib-engines.xml#
SIBMQLink_1132607756126)'
```

Create a SIBMQLink that uses queue points on the same messaging engine as the target destination, whenever possible.

```
AdminTask.createSIBMQLink('[-bus bus1 -messagingEngine cluster1.000-bus1
-name MyMQLink -foreignBusName -MQQMgr1 -queueManagerName bus1
-senderChannelTransportChain OutboundBasicMQLink  -preferLocal TRUE]')
```

deleteSIBMQLink command

Use the deleteSIBMQLink command to delete a WebSphere MQ link from a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The deleteSIBMQLink command deletes a WebSphere MQ link from a bus.

CAUTION:
This command deletes the WebSphere MQ link and its additional property configuration. Although there is no confirmation prompt before deleting the WebSphere MQ link, the link is not deleted from the master configuration until you save your changes.

Target object

A bus.

Required parameters

-bus
The name of the service integration bus for which you want to delete a WebSphere MQ link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine to which the WebSphere MQ link was assigned when it was created. You can use the listSIBEngines command to list the names of existing messaging engines.

-mqLink
The name of the WebSphere MQ link that you want to delete.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.deleteSIBMQLink('[-bus bus1 -messagingEngine cluster1.000-bus1
-mqLink myMQLink2]')
```

listSIBMQLinks command

Use the listSIBMQLinks command to list all WebSphere MQ links for a specified service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

The listSIBMQLinks command lists the names of all WebSphere MQ links for a specified service integration bus. A WebSphere MQ link connects a messaging engine as a queue manager to WebSphere MQ, thereby providing a bridge between a service integration bus and a WebSphere MQ network.

Target object

A specified service integration bus.

Required parameters

-bus
The name of the service integration bus for which you want to list WebSphere MQ links.

Conditional parameters

None.

Optional parameters

-node
The name of the node for which you want to list WebSphere MQ links. This option restricts the list of WebSphere MQ links to those links assigned to the node.

-server
The name of the server for which you want to list WebSphere MQ links. This option restricts the list of WebSphere MQ links to those links assigned to the server.

-cluster
The name of the server cluster for which you want to list WebSphere MQ links. This option restricts the list of WebSphere MQ links to those links assigned to the server cluster.

-messagingEngine
The name of the messaging engine for which you want to list WebSphere MQ links. This option restricts the list of WebSphere MQ links to those links assigned to the messaging engine.

Example

```
AdminTask.listSIBMQLinks('[-bus bus1 -foreignBus foreignbus1 ]')
'foreignbus1(cells/cell01/nodes/node01/servers/server1|sib-engines.xml#
SIBMQLink_1213002780841)'
```

modifySIBMQLink command

Use the modifySIBMQLink command to modify the properties of a WebSphere MQ link.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The modifySIBMQLink command modifies the properties of a WebSphere MQ link.

Target object

A WebSphere MQ link.

Required parameters

-bus
The name of the service integration bus for which you created the WebSphere MQ link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine for which you created the WebSphere MQ link. You can use the listSIBEngines command to list the names of existing messaging engines.

-name
The name of the WebSphere MQ link.

Conditional parameters

None.

Optional parameters

WebSphere MQ link configuration parameters:

-queueManagerName
The name of the virtual queue manager associated with the messaging engine, and by which the messaging engine is known to a remote WebSphere MQ network.

The queue manager name must conform to the WebSphere MQ naming conventions; for example, the name must be a maximum of 48 characters.

-description
An optional description for the bus, for administrative purposes.

-exceptionDestination
The destination for an inbound message when the WebSphere MQ link cannot deliver the message to its target bus destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist.

System
The WebSphere MQ link uses the default exception destination. All messages that cannot be delivered to the bus destination are rerouted automatically to the system default exception

destination for the messaging engine that this link is assigned to: _SYSTEM.Exception.Destination*messaging_engine_name*.

None The WebSphere MQ link has no exception destination. Undeliverable messages are not rerouted to an exception destination and can block the processing of other messages waiting for delivery through the link to the same bus. This option can be used to preserve message ordering.

Specify
The WebSphere MQ link uses the exception destination specified here. If this is not possible, it uses the system exception destination.

-batchSize
The maximum number of messages that can be sent through a channel before taking a checkpoint.

The batch size does not affect the way the sender and receiver channels for this link transfer messages. Messages are always transferred individually, but are committed or backed out as a batch.

For more information about choosing the batch size, see the description of the batch size (**BATCHSZ**) property in the *Intercommunication* section of the WebSphere MQ information center.

Default	50
Range	1 through 9999

-maxMsgSize
The maximum message length, in bytes, that can be transmitted on any channel for the WebSphere MQ link. This is compared with the value for the corresponding partner WebSphere MQ channel and the actual maximum used is the lower of the two values.

For information about how to choose an appropriate number, see the description of the Maximum message length (**MAXMSGL**) property in the *Intercommunication* section of the WebSphere MQ information center.

Default	4194304 bytes (4MB)
Range	0 through 104857600 Specify 0 to use the largest value that the target queue manager will honor.

-heartBeat
The negotiated time, in seconds, between heartbeat flows passed from the WebSphere MQ link sender channel to the WebSphere MQ receiver channel when there are no messages on the transmission point being served by the WebSphere MQ link sender channel.

Heartbeats give the receiving channel the opportunity to quiesce the channel connection.

For more information about choosing the value for this property, see the *Intercommunication* section of the WebSphere MQ information center.

Default	300 seconds
Range	0 through 999999

-sequenceWrap
The value at which message sequence numbers wrap to start again at 1. For example, if you specify a value of 1000, when the message sequence number reaches 1001 it will restart at 1.

For more information about choosing the value for this property, see the description of the Sequence Number wrap (**SEQWRAP**) property in the *Intercommunication* section of the WebSphere MQ information center.

Default	999999999

Range 100 through 999999999

-nonPersistentMessageSpeed Fast | Normal
The class of service for nonpersistent messages on channels of this WebSphere MQ link.

Default Fast
Range

Fast Nonpersistent messages can be lost if there is a transmission failure or if the channel stops when the messages are in transit.

Normal
Nonpersistent messages are not lost if there is a transmission failure or if the channel stops when the messages are in transit.

-adoptable True | False
A property of the WebSphere MQ link, which shows whether a running instance of a WebSphere MQ link receiver channel (associated with this WebSphere MQ link) should be adopted. In the event of a communications failure, a running instance of a WebSphere MQ link receiver channel might be left waiting for messages. When communication is reestablished, and the partner WebSphere MQ sender channel next attempts to establish a session with the WebSphere MQ link receiver channel, the request will fail as there is already a running instance of the WebSphere MQ link receiver channel that believes it is in session with the partner WebSphere MQ sender channel. You can overcome this problem by selecting this option, which causes the already running instance of the WebSphere MQ link receiver channel to be stopped and a new instance to be started.

If you set this option to True, the WebSphere MQ sender channels might reestablish a connection to this WebSphere MQ link in the event that a communications failure has occurred and the link has not yet detected the failure condition.

Default True

-initialState Started | Stopped
The state of the WebSphere MQ link, which shows whether the link is started or stopped when the hosting messaging engine is first started. Until it is started, the WebSphere MQ link is unavailable.

Default Started
Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

WebSphere MQ link sender channel parameters:

-senderChannelName
The sender channel that sends messages to the gateway queue manager. The sender channel communicates with a WebSphere MQ receiver channel on the gateway queue manager, and converts service integration bus messages to MQ format messages.

This name must be the same as the name of the receiver channel on WebSphere MQ.

For more information about choosing channel names, see the description of the channel name (**CHANNEL**) property in the *Intercommunication* section of the WebSphere MQ information center.

-hostName
The host name or TCP/IP address for the gateway queue manager that is used to connect into the WebSphere MQ network.

Type the host name or IP address of the host on which the gateway queue manager runs.

If this field is blank, the gateway queue manager is assumed to be running on the same host as the messaging engine on which the WebSphere MQ link is defined.

-port
The TCP/IP port number on which the gateway queue manager is listening for the WebSphere MQ link.

Default	1414
Range	0 through 65535

-connameList
The connection name list for the gateway queue manager which is used to connect to the WebSphere MQ network. The connections are tried in the order in which they are specified in the connection name list until a connection is successfully established. If no connection is successful, the channel starts retry processing.

The connection names must be given as a comma separated list in the following format:

```
Dnsname1(portnumber1) , Dnsname2(portnumber2)
```

Type the DNS name or the IP address of the host on which the gateway queue manager is running. Type the port number in the range 0 through 65535.

If you do not specify the connection name list parameter, the gateway queue manager is assumed to be running on the same host as the messaging engine on which the WebSphere MQ link is defined.

If you do specify the connection list parameter, you must have already specified a value for the sender channel name parameter, otherwise the value that you specify for the connection name list parameter is ignored.

If you specify the host name and port parameter as well as the connection name list parameter, then at runtime the connection name list takes precedence and host name and port values are ignored.

-discInterval
The time in seconds for which the sender channel waits for new messages to arrive on the transmission queue after sending a batch of messages. The channel disconnects after this interval, and must be restarted manually or by triggering.

The default value is a reasonable interval. Change this value only if you understand the implications for performance, and you need a different value for the requirements of the traffic flowing down your channels.

Performance is affected by the value specified for the disconnect interval. A very low value (a few seconds) can cause an unacceptable amount of processing in constantly starting up the channel. A very large value (more than an hour) might mean that system resources are unnecessarily held up.

If you want your channels to be active only when there are messages for them to transmit, you should set the disconnect interval to a fairly low value. Note that the default setting is quite high and so is not recommended for channels where this level of control is required. Because it is difficult to interrupt the receiving channel, the most economical option is to have the channel automatically disconnect and reconnect as the workload demands. For most channels, the appropriate setting of the disconnect interval can be established heuristically.

Default	900 seconds

Range 0 through 999999

A value of 0 (zero) means never disconnect; the channel waits indefinitely for messages.

-shortRetryCount

The maximum number of times that the sender channel tries to restart after a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then the long retry mechanism is invoked.

Default 10
Range 0 through 999999999

-shortRetryInterval

The number of seconds between attempts by the sender channel to restart after a communication or partner failure.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Default 60 seconds
Range 0 through 999999

-longRetryCount

The maximum number of times that the sender channel tries to restart after the short retry mechanism did not recover from a communication or partner failure. If the connection name list is provided, during each retry the connections are tried in the order in which they are specified in the connection list until a connection is successfully established. If the count of remaining retries reaches zero, and the channel has not restarted, then an error is logged and the channel is stopped.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Default 999999999
Range 0 through 999999999

-longRetryInterval

The number of seconds between attempts by the sender channel to restart after the short retry mechanism did not recover from a communication or partner failure.

For more information about using retry mechanisms with WebSphere MQ, see the *Intercommunication* section of the WebSphere MQ information center.

Default 1200 seconds
Range 0 through 999999

-senderChannelInitialState Started | Stopped

The state of the WebSphere MQ link, which shows whether the sender channel is started or stopped when the associated WebSphere MQ link is first started. Until it is started, the channel is unavailable.

Default Started

Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

WebSphere MQ link receiver channel parameters:

-receiverChannelName
The name of the receiver channel for the WebSphere MQ link, used to receive messages from WebSphere MQ onto the bus.

This name must be the same as the name of the sender channel on WebSphere MQ.

-inboundNonPersistentReliability Best effort | Express | Reliable
The acceptable reliability of message delivery for nonpersistent message flows from WebSphere MQ through this WebSphere MQ link, from Best effort to Reliable, in order of increasing reliability.

This reliability delivery option is assigned to all WebSphere MQ nonpersistent messages flowing over this receiver channel.

Default
Reliable

Range

Best effort
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Express
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Reliable
Messages are discarded when a messaging engine stops or fails.

-inboundPersistentReliability Reliable | Assured
The acceptable reliability of message delivery for inbound persistent message flows from WebSphere MQ through this WebSphere MQ link, from Reliable to Assured, in order of increasing reliability.

Default
Assured

Range

Reliable
Messages might be discarded when a messaging engine fails.

Assured
Messages are not discarded.

-receiverChannelInitialState Started | Stopped

The state of the WebSphere MQ link, which shows whether the receiver channel is started or stopped when the associated WebSphere MQ link is first started. Until it is started, the channel is unavailable.

Default
Started

Range

Stopped
When the associated messaging engine is started, the WebSphere MQ link is in a stopped state and cannot communicate with the WebSphere MQ network.

Started
When the associated messaging engine is started, the WebSphere MQ link is started automatically and is enabled for communication with the WebSphere MQ network.

-preferLocal TRUE | FALSE

This option indicates that the link prefers to send incoming messages to the queue point of the target destination that is located on the same messaging engine as the link, if available. The link must be owned by a messaging engine running on a WebSphere Application Server Version 7.0 or later server.

This option is supported only when used by a JMS application that is running with a WebSphere Application ServerVersion 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application ServerVersion 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Default
TRUE

Range

TRUE Send each incoming message to the queue point of the target destination that is located on the same messaging engine as the link, if available.

FALSE Send incoming messages to any queue points of the target destinations.

Examples

Show the properties of a WebSphere MQ link, modify the link to change the batch size, then show the properties of the link again.

```
wsadmin>AdminTask.showSIBMQLink('[-bus myBus -messagingEngine myHostNode01.server1-myBus
-mqLink myMQLink]')
'{nonPersistentMessageSpeed=FAST, qmName=myBus, adoptable=false, sequenceWrap=999999999,
 name=myMQLink, targetUuid=738AE126B908E5451A3D4691, initialState=STARTED,
 senderChannel=null, brokerProfile=[], receiverChannel=null, preferLocalQueuePoints=true,
 batchSize=50, uuid=6B89C4F08AB072C5, heartBeat=300, description=null, maxMsgSize=4194304,
 exceptionDestination=_SYSTEM.Exception.Destination.myHostNode01.server1-myBus}'
wsadmin>
wsadmin>AdminTask.modifySIBMQLink('[-bus myBus -messagingEngine myHostNode01.server1-myBus
-name myMQLink -batchSize 100]')
'myMQLink(cells/cell01/nodes/node01/servers/server1|sib-engines.xml#SIBMQLink_1132608724468)'

wsadmin>AdminTask.showSIBMQLink('[-bus myBus -messagingEngine myHostNode01.server1-myBus
-mqLink myMQLink]')
'{nonPersistentMessageSpeed=FAST, qmName=myBus, adoptable=false, sequenceWrap=999999999,
 name=myMQLink, targetUuid=738AE126B908E5451A3D4691, initialState=STARTED,
 senderChannel=null, brokerProfile=[], receiverChannel=null, preferLocalQueuePoints=true,
 batchSize=100, uuid=6B89C4F08AB072C5, heartBeat=300, description=null, maxMsgSize=4194304,
 exceptionDestination=_SYSTEM.Exception.Destination.myHostNode01.server1-myBus}'
wsadmin>
```

Modify a WebSphere MQ link so that the link has no preference for which queue points to use.

```
wsadmin>AdminTask.modifySIBMQLink('[-bus Bus1 -messagingEngine
node1.node1server1-Bus1 -name MQLink -queueManagerName QM2 -preferLocal FALSE]')
wsadmin>
```

showSIBMQLink command

Use the showSIBMQLink command to show the properties of a WebSphere MQ link.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

The showSIBMQLink command lists details about the properties of a WebSphere MQ link.

Target object

A WebSphere MQ link.

Required parameters

-bus
The name of the service integration bus for which you created the WebSphere MQ link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine for which you created the WebSphere MQ link. You can use the listSIBEngines command to list the names of existing messaging engines.

-name
The name of the WebSphere MQ link.

Conditional parameters

None.

Optional parameters

javaFormat
The output from the command is a format suitable for Java program clients.

Example

```
wsadmin>AdminTask.showSIBMQLink('[-bus myBus -messagingEngine myHostNode01.server1-myBus
-mqLink myMQLink]')
'{nonPersistentMessageSpeed=FAST, qmName=myBus, adoptable=false, sequenceWrap=999999999,
name=myMQLink, targetUuid=738AE126B908E5451A3D4691, initialState=STARTED,
senderChannel=null, brokerProfile=[], receiverChannel=null, preferLocalQueuePoints=true,
batchSize=50, uuid=6B89C4F08AB072C5, heartBeat=300, description=null, maxMsgSize=4194304,
exceptionDestination=_SYSTEM.Exception.Destination.myHostNode01.server1-myBus}'
wsadmin>
```

SIBAdminCommands: Bus link administrative commands for the AdminTask object

You can use these administrative commands to manage service integration bus links. A service integration bus link defines a link between a messaging engine in one service integration bus and a messaging engine in a foreign service integration bus.

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and Jython.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

createSIBLink command

Use the createSIBLink command to create a new service integration bus link.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The createSIBLink command creates a new bus link for a specified bus. A bus link defines a link between a messaging engine in one bus and a messaging engine in a foreign bus.

Target object

None.

Required parameters

-bus
The name of the service integration bus for which you want to create the service integration bus link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine for which you want to create the service integration bus link. The service integration bus link connects a messaging engine to another messaging engine in a different service integration bus, thereby providing a bridge between two service integration buses.

-name
The name by which you want the service integration bus link to be known.

-foreignBusName
The name of the foreign bus that defines the remote service integration bus for the service integration bus link. You can use the listSIBForeignBuses command to list the names of existing foreign buses.

-bootstrapEndpoints
The comma-separated list of endpoints used to connect to a bootstrap server.

This property is set in the same way as the **Provider endpoint** property in the JMS connection factory settings. For more information, see the steps relating to setting bootstrap endpoints in “Configuring a connection to a non-default bootstrap server” on page 531.

The port for the bootstrap endpoint is the port defined on the SIB endpoint address that is configured on the target application server on the foreign bus.

-remoteMessagingEngineName
The messaging engine on the foreign bus to which this link connects.

Conditional parameters

None.

Optional parameters

-description
An optional description for the bus, for administrative purposes.

-protocolName
The type of transport chain used for communication with the foreign bus.

The transport chain name must be the name of the transport chain as defined on the server on which the target messaging engine is hosted.

-authAlias
The name of the authentication alias, used to authenticate access to the foreign bus.

You must have predefined a J2C authentication alias.

-exceptionDestination
The destination for an inbound message when the service integration bus link cannot deliver the message to its target bus destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist.

System
The service integration bus link uses the default exception destination. All messages that cannot be delivered to the bus destination are rerouted automatically to the system default

exception destination for the messaging engine that this link is assigned to: _SYSTEM.Exception.Destination.*messaging_engine_name*.

None The service integration bus link has no exception destination. Undeliverable messages are not rerouted to an exception destination and can block the processing of other messages waiting for delivery to the same destination. This option can be used to preserve message ordering.

Specify
The service integration bus link uses the exception destination specified here. If this is not possible, it uses the system exception destination.

-initialState Started | Stopped
The state of the gateway link, which shows whether the link is started automatically when the messaging engine is started.

Default Started

Range

Stopped
When the associated messaging engine is started, the gateway link is in a stopped state and cannot process any new requests for connections.

Started
When the associated messaging engine is started, the gateway link is in a started state and can process any new requests for connections.

-preferLocal TRUE | FALSE
Indicates that the link prefers to send incoming messages to the queue point of the target destination that is located on the same messaging engine as the link, if available. The link must be owned by a messaging engine running on a WebSphere Application Server Version 7.0 or later server.

This option is supported only when used by a JMS application that is running with a WebSphere Application ServerVersion 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application ServerVersion 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Default TRUE

Range

TRUE Send each incoming message to the queue point of the target destination that is located on the same messaging engine as the link, if available.

FALSE Send incoming messages to any queue points of the target destinations.

Examples

```
AdminTask.createSIBLink('[-bus bus1 -messagingEngine node01.server1-myBus
-name mySIBLink -foreignBusName bus2 -bootstrapEndpoints host1:1111:chain1
-remoteMessagingEngineName node02.server2-bus2]')
```

Create a service integration bus link that uses a queue point on the same messaging engine as the link, whenever possible.

```
AdminTask.createSIBLink('[-bus bus1 -messagingEngine node01.server1-bus1
-name mySIBLink -foreignBusName bus2 -bootstrapEndpoints host1:1111:chain1
-remoteMessagingEngineName node02.server2-bus2 -preferLocal TRUE]')
```

deleteSIBLink command

Use the deleteSIBLink command to delete a service integration bus link from a bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The deleteSIBLink command deletes a bus link from a bus.

CAUTION:
This command deletes the service integration bus link and its additional property configuration. Although there is no confirmation prompt before deleting the service integration bus link, the link is not deleted from the master configuration until you save your changes.

Target object

A bus.

Required parameters

-bus
The name of the service integration bus for which you want to delete a service integration bus link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine to which the service integration bus link was assigned when created. You can use the listSIBEngines command to list the names of existing messaging engines.

-sibLink
The name of the service integration bus link that you want to delete.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.deleteSIBLink('[-bus bus1 -messagingEngine node01.server1-bus1 -sibLink mySIBLink]')
```

listSIBLinks command

Use the listSIBLinks command to list all service integration bus links for a given bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

The listSIBLinks command lists the names of all service integration bus links for a specified service integration bus. A service integration bus link defines a link between a messaging engine in one service integration bus and a messaging engine in a foreign service integration bus.

Target object

A bus.

Required parameters

-bus
The name of the service integration bus for which you want to list service integration bus links.

Conditional parameters

None.

Optional parameters

-node
The name of the node for which you want to list service integration bus links. This option restricts the list of service integration bus links to those links assigned to the node.

-server
The name of the server for which you want to list service integration bus links. This option restricts the list of service integration bus links to those links assigned to the server.

-messagingEngine
The name of the messaging engine for which you want to list service integration bus links. This option restricts the list of service integration bus links to those links assigned to the messaging engine.

Example

```
AdminTask.listSIBLinks('[-bus bus1 -foreignBus bus2]')
'mySIBLink(cells/cell01/nodes/node01/servers/server1|sib-engines.xml#
SIBLink_1212163147845)'
wsadmin>
```

modifySIBLink command

Use the modifySIBLink command to modify the properties of a service integration bus link.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The modifySIBLink command modifies the properties of a service integration bus link.

Target object

None.

Required parameters

-bus
The name of the service integration bus for which you want to modify the service integration bus link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine for which you want to modify the service integration bus link. The service integration bus link connects a messaging engine to another messaging engine in a different service integration bus, thereby providing a bridge between two service integration buses.

-name
The name by which you want the service integration bus link to be known.

-foreignBusName
The name of the foreign bus that defines the remote service integration bus for the service integration bus link. You can use the listSIBForeignBuses command to list the names of existing foreign buses.

Conditional parameters

None.

Optional parameters

-bootstrapEndpoints
The comma-separated list of endpoints used to connect to a bootstrap server. This property is set in

the same way as the **Provider endpoint** property in the JMS connection factory settings. For more information, see the steps relating to setting bootstrap endpoints in “Configuring a connection to a non-default bootstrap server” on page 531.

The port for the bootstrap endpoint is the port defined on the SIB endpoint address that is configured on the target application server on the foreign bus.

-remoteMessagingEngineName
The messaging engine on the foreign bus to which this link connects.

-description
An optional description for the bus, for administrative purposes.

-protocolName
The type of transport chain used for communication with the foreign bus.

The transport chain name must be the name of the transport chain as defined on the server on which the target messaging engine is hosted.

-authAlias
The name of the authentication alias, used to authenticate access to the foreign bus.

You must have predefined a J2C authentication alias.

-exceptionDestination
The destination for an inbound message when the service integration bus link cannot deliver the message to its target destination, or to the exception destination that is configured for that target destination, or when the target destination does not exist.

System
The service integration bus link uses the default exception destination. All messages that cannot be delivered to the bus destination are rerouted automatically to the system default exception destination for the messaging engine that this link is assigned to: _SYSTEM.Exception.Destination.*messaging_engine_name*.

None The service integration bus link has no exception destination. Undeliverable messages are not rerouted to an exception destination and can block the processing of other messages waiting for delivery to the same destination. This option can be used to preserve message ordering.

Specify
The service integration bus link uses the exception destination specified here. If this is not possible, it uses the system exception destination.

-initialState Started | Stopped
The state of the gateway link, which shows whether the link is started automatically when the messaging engine is started.

Default
Started

Range

Stopped
When the associated messaging engine is started, the gateway link is in a stopped state and cannot process any new requests for connections.

Started
When the associated messaging engine is started, the gateway link is in a started state and can process any new requests for connections.

-preferLocal TRUE | FALSE
Indicates whether the link prefers to send incoming messages to the queue point of the target destination that is located on the same messaging engine as the link, if available. The link must be owned by a messaging engine running on a WebSphere Application Server Version 7.0 or later server.

This option is supported only when used by a JMS application that is running with a WebSphere Application ServerVersion 7.0 or later server or client, and that is connected to a messaging engine running on WebSphere Application ServerVersion 7.0 or later server. Use on previous versions of WebSphere Application Server will result in an exception to the application.

Default TRUE

Range

TRUE Send each incoming message to the queue point of the target destination that is located on the same messaging engine as the link, if available.

FALSE Send incoming messages to any queue points of the target destinations.

Examples

```
wsadmin>AdminTask.showSIBLink('[-bus bus1 -messagingEngine node01.server1-bus1  -sibLink
mySIBLink]')
'{bootstrapEndpoints=host1:1111:chain1, protocolName=null, authAlias=null,
preferLocalQueuePoints=true, name=mySIBLink, uuid=34647E59163B253D,
remoteMessagingEngineName=node02.server2-bus2, description=null,
targetUuid=BAD49BA75CD36D740E366978, initialState=STARTED,
exceptionDestination=$DEFAULT_EXCEPTION_DESTINATION}'

wsadmin>AdminTask.modifySIBLink('[-bus bus1 -messagingEngine node01.server1-bus1
-name MySIBLink -foreignBusName -bus2
-bootstrapEndpoints anotherhost:2222:BootstrapBasicMessaging]')

wsadmin>AdminTask.showSIBLink('[-bus bus1 -messagingEngine node01.server1-bus1  -sibLink
mySIBLink]')
'{bootstrapEndpoints=anotherhost:2222:BootstrapBasicMessaging, protocolName=null,
authAlias=null, preferLocalQueuePoints=true, name=mySIBLink, uuid=34647E59163B253D,
remoteMessagingEngineName=node02.server2-bus2, description=null,
targetUuid=BAD49BA75CD36D740E366978, initialState=STARTED,
exceptionDestination=$DEFAULT_EXCEPTION_DESTINATION}'
```

Modify a service integration bus link so that there is no preference for which queue points to use.

```
AdminTask.modifySIBLink('[-bus bus1 -messagingEngine cluster1.000-bus1 -name MyLink
-foreignBusName -FB1 bootstrapEndpoints host1:1111:chain1 -remoteMessagingEngineName
-cluster2.000-FB1 -preferLocal FALSE]')
```

showSIBLink command

Use the showSIBLink command to show the properties of a service integration bus link.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('`*command_name*`')`

Purpose

The showSIBLink command lists details about the properties of a service integration bus link.

Target object

A bus link.

Required parameters

-bus
The name of the service integration bus for which you created the service integration bus link. You can use the listSIBuses command to list the names of existing buses.

-messagingEngine
The name of the messaging engine for which you created the service integration bus link. You can use the listSIBEngines command to list the names of existing messaging engines.

-sibLink
The name of the service integration bus link.

Conditional parameters

None.

Optional parameters

None.

Example

```
AdminTask.showSIBLink('[-bus bus1 -messagingEngine node01.server1-bus1
-sibLink mySIBLink]')
'{bootstrapEndpoints=host1:1111:chain1, protocolName=null, authAlias=null,
preferLocalQueuePoints=true, name=aSIBLink, uuid=34647E59163B253D,
remoteMessagingEngineName=wasinstallNode01.server1-MQServerBus, description=null,
targetUuid=BAD49BA75CD36D740E366978, initialState=STARTED,
exceptionDestination=_SYSTEM.Exception.Destination.node01.server1-bus1}'
```

SIBAdminCommands: Messaging engine administrative commands for the AdminTask object

You can use these administrative commands to manage messaging engines.

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and Jython.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

listSIBEngines command

Use the listSIBEngines command to list the messaging engines for a service integration bus member.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all the messaging engines on a bus member.

Target object

A bus member.

Required parameters

-bus *busname*
The name of the service integration bus on which the bus member is configured. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

-node *nodename* **-server** *servername*
To list the messaging engines for an application server as a bus member, specify both the name of the node on which the server runs and the name of the server.

-cluster *cluster*
To list the messaging engines for a server cluster as a bus member, specify the name of the cluster.

This option should be used only in WebSphere Application Server environments that support server clusters.

Optional parameters

None.

Example

```
AdminTask.listSIBEngines('[-bus bus1 ]')
'node01.server1-bus1(cells/cell01/nodes/node01/servers/server1|sib-engines.xml#
SIBMessagingEngine_1212163145962)\r\n
node02.server2-bus2(cells/cell01/nodes/node02/servers/server2|sib-engines.xml#
SIBMessagingEngine_1212163146273)'
```

modifySIBEngine command

Use the modifySIBEngine command to modify the properties of a messaging engine for a service integration bus member.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command modifies the properties of a messaging engine.

Target object

A messaging engine.

Required parameters

-bus *busname*
The name of the service integration bus on which the bus member is configured. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

-node *nodename* **-server** *servername*
To change properties of a messaging engine for an application server as a bus member, specify both the name of the node on which the server runs and the name of the server.

-cluster *cluster*
To change properties of a messaging engine for a server cluster as a bus member, specify the name of the cluster.

This option should be used only in WebSphere Application Server environments that support server clusters.

-engine *enginename*
If the bus member has only one messaging engine, you do not need to specify the engine name. If the bus member has several messaging engines, you must specify the name of the engine for which you want to change properties.

Optional parameters

-description *text*
An optional description for the messaging engine, for administrative purposes.

-initialState STARTED | STOPPED
The initial state determines whether the messaging engine is started automatically when the server is started.

Stopped
When the associated application server is started, the messaging engine is stopped and is not available to process messages.

Started
When the associated application server is started, the messaging engine is started and is available to process messages.

-highMessageThreshold *number*
The maximum total number of messages that the messaging engine can place on its message points.

When the messaging engine is created, the high message threshold of the bus is used to set the default value for this property. When a message point is created on this messaging engine, the value of this property is used to set the default high message threshold for the message point.

Example

```
wsadmin>AdminTask.showSIBEngine('[-bus bus1 -node node01 -server server1
-engine node01.server1-bus1 ]')
'{initialState=STARTED, targetGroups=[], name=node01.server1-bus1,
highMessageThreshold=50000, messageStoreType=FILESTORE, uuid=56F8FE11AB84188D,
busName=bus1, busUuid=6DF19B02BC879BD1}'

wsadmin>AdminTask.modifySIBEngine('[-bus bus1 -node node01 -server server1
-engine node01.server1-bus1 -initialState STOPPED ]')

wsadmin>AdminTask.showSIBEngine('[-bus bus1 -node node01 -server server1
-engine node01.server1-bus1 ]')
'{initialState=STOPPED, targetGroups=[], name=node01.server1-bus1,
highMessageThreshold=50000, messageStoreType=FILESTORE, uuid=56F8FE11AB84188D,
busName=bus1, busUuid=6DF19B02BC879BD1}'
```

showSIBEngine command

Use the showSIBEngine command to list properties of a messaging engine for a service integration bus member.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

This command lists details about properties of a messaging engine.

Target object

A bus member.

Required parameters

-bus *busname*
The name of the service integration bus on which the bus member is configured. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

-node *nodename* **-server** *servername*
To list properties of a messaging engine for an application server as a bus member, specify both the name of the node on which the server runs and the name of the server.

-cluster *cluster*
To list properties of a messaging engine for a server cluster as a bus member, specify the name of the cluster.

This option should be used only in WebSphere Application Server environments that support server clusters.

-engine *enginename*
If the bus member has only one messaging engine, you do not need to specify the engine name. If the bus member has several messaging engines, you must specify the name of the engine for which you want to display details.

Optional parameters

None.

Example

```
AdminTask.showSIBEngine('[-bus bus1 -node node01 -server server1
-engine node01.server1-bus1 ]')
'{initialState=STARTED, targetGroups=[], name=node01.server1-bus1,
highMessageThreshold=50000, messageStoreType=FILESTORE, uuid=56F8FE11AB84188D,
busName=bus1, busUuid=6DF19B02BC879BD1}'
```

deleteSIBEngine command

Use the deleteSIBEngine command to delete a messaging engine from a service integration bus member.

You should be wary of deleting and recreating messaging engines on bus members for which WS-Notification-administered subscribers have been configured, because in some cases this can leave the remote web service subscription active (and passing notification messages to the local server) even though there is no longer any record of it. For more information, see the WS-Notification troubleshooting tip Problems can occur when deleting administered subscribers and messaging engines.

If you promote a server bus member to a cluster that is not a member of the bus, do not delete then recreate the messaging engine. Use the migrateServerMEtoCluster command instead.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes a messaging engine from a bus member. Any associated configurations of the messaging engine, such as core group policies or destinations, are also deleted. If the messaging engine uses a data store for the message store, objects in the data stores remain so that you can still access them. If you recreate the same messaging engine, you must remove any old data store tables before you start the new messaging engine.

This command also cleans up any mediation execution points that are on the messaging engine as the result of mediating a destination to a WebSphere MQ server bus member. The command unmediates the destination to which the mediation execution point corresponds.

Target object

A messaging engine.

Required parameters

-bus *bus_name*
The name of the service integration bus on which the bus member is configured. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

-node *node_name*
-server *server_name*
To delete a messaging engine from an application server that is a bus member, specify both the name of the node on which the server runs and the name of the server.

-engine *engine_name*
If the bus member has only one messaging engine, you do not need to specify the engine name. If the bus member has several messaging engines, you must specify the name of the engine that you want to delete.

Optional parameters

None.

Example

Delete the messaging engine from server1 on node1 that is a member of a bus1.

```
AdminTask.deleteSIBEngine ('[-bus bus1 -node node1 -server server1]')
```

SIBAdminCommands: Destination administrative commands for the AdminTask object

You can use these administrative commands to manage bus destinations.

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and JACL.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

createSIBDestination command

Use the createSIBDestination command to create a new bus destination for a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The createSIBDestination command creates a new bus destination for a service integration bus. With this command, a messaging destination can also be localized to a WebSphere MQ server bus member.

Target object

A service integration bus.

Required parameters

-bus
The name of the service integration bus on which to create the bus destination. To list the names of existing buses, use the listSIBuses command.

-name
The identifier by which this destination is known for administrative purposes.

-type
Indicates the type of bus destination that you want to create:

Alias An alias destination, that provides a level of abstraction between applications and the underlying target bus destinations that hold messages. Applications interact with the alias destination, so the target bus destination can be changed without changing the application.

Foreign
A foreign destination, which identifies a destination on another bus, and enables applications on one bus to access the destination on another bus directly.

Port Represents a particular message and transport binding for an outbound service that communicates with an externally-hosted target web service.

Queue
A queue, for point-to-point messaging.

TopicSpace
A topic space, for publish/subscribe messaging.

WebService
Represents an externally-hosted target web service.

Conditional parameters

None.

Optional parameters

-cluster
To assign the bus destination to a cluster bus member, specify the name of the cluster. Do not specify the **-node**, **-server** or **-wmqServer** parameters.

-node
To assign the bus destination to a server bus member, specify both the name of the node on which the server runs and the name of the server. Do not specify the **-cluster** or **-wmqServer** parameters.

-server
To assign the bus destination to a server bus member, specify both the name of the node on which the server runs and the name of the server. Do not specify the **-cluster** or **-wmqServer** parameters.

-wmqServer
To assign the bus destination to a WebSphere MQ queue, specify both the name of the WebSphere MQ server bus member where the destination is assigned (this parameter), and the name of the WebSphere MQ queue used to store messages sent to the destination (the **-wmqQueueName** parameter). Set the**-wmqServer** parameter to the name you gave when you created the WebSphere MQ server. Set the **-wmqQueueName** parameter to the name allocated to the WebSphere MQ queue by WebSphere MQ administration. Do not specify the **-cluster**, **-node** or **-server** parameters.

-aliasBus
If you are creating an alias destination, specify the source bus name of the alias mapping.

-targetBus
If you are creating an alias destination, specify the name of the bus to which the alias destination is mapped.

-targetName
If you are creating an alias destination, specify the name of the destination to which the alias destination is mapped.

-foreignBus
If you are creating a foreign destination, specify the name of the foreign bus.

-description
Specify a description for the bus destination, for administrative purposes.

-reliability
Specify the default reliability level to assign to a message produced to this destination when an explicit reliability has not been set by the producer application. Service integration supports five reliability levels (also known as delivery options or qualities of service):

BEST_EFFORT_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

EXPRESS_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

RELIABLE_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails.

RELIABLE_PERSISTENT
Messages might be discarded when a messaging engine fails.

ASSURED_PERSISTENT
Messages are not discarded.

Note: Higher levels of reliability have higher impacts on performance.

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

-maxReliability
Specify the maximum reliability level that is accepted for values specified by producer applications. Service integration supports five reliability levels (also known as delivery options or qualities of service):

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

-nonPersistentReliability

Specify the service integration quality of service to use with nonpersistent WebSphere MQ messages that are received by service integration from a WebSphere MQ network. The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For nonpersistent WebSphere MQ messages received, the default service integration quality of service is RELIABLE_NONPERSISTENT. If you choose to override this default, you will probably choose one of the other nonpersistent service integration qualities of service BEST_EFFORT_NONPERSISTENT or EXPRESS_NONPERSISTENT. However, you can choose any of the five possible service integration qualities of service:

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

For more information, see ../ae/rjc0014_.dita.

-persistentReliability

Specify the service integration quality of service to use with persistent WebSphere MQ messages that are received by service integration from a WebSphere MQ network. The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For persistent WebSphere MQ messages received, the default service integration quality of service is ASSURED_PERSISTENT. If you choose to override this default, you will probably choose the other persistent service integration quality of service RELIABLE_PERSISTENT. However, you can choose any of the five possible service integration qualities of service:

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

For more information, see ../ae/rjc0014_.dita.

-overrideOfQOSByProducerAllowed TRUE | FALSE

Controls the quality of service for message flows between producers and the destination. Select this option to use the quality of service specified by producers instead of the quality defined for the destination.

-defaultPriority *number*

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

-maxFailedDeliveries *number*

The maximum number of failed attempts to process a message. After this number of failed attempts, the message is forwarded from the intended destination to its exception destination. Specify a value in the range 0 through 2147483647. A value of 0 (zero) means that if a message cannot be delivered on the first attempt, it is either forwarded to the exception destination or discarded, as defined by the **-exceptionDestination** parameter.

-exceptionDestination *value*

Use these properties to define what happens to any messages that cannot be delivered to this destination.

By default, all messages that cannot be delivered to this destination are rerouted to the system default exception destination for the messaging engine to which this destination is assigned (_SYSTEM.Exception.Destination.*messaging_engine_name*). Use this parameter to override the default value. You can set a specific exception destination for this destination, or you can specify that undeliverable messages are not rerouted to an exception destination by entering an empty string (""), in which case the maximum failed deliveries count has no effect.

Note: An undeliverable message can block the processing of other messages waiting for delivery to the same destination.

You can use this option and specify no exception destination to preserve message ordering.

-sendAllowed TRUE | FALSE
Clear this option (setting it to FALSE) to stop producers from being able to send messages to this destination.

- For a queue point of a non-mediated destination, or a mediation point of a mediated destination, if you clear this option then new messages (from attached producers or forwarded from another destination) are redirected to any available message point. If no message points are available, then messages that have already been accepted onto the bus, and new messages from attached producers, are preserved by the bus until a message point becomes available. The only exception to this is the case of a destination with only one message point (queue point or mediation point depending on whether the destination is mediated or non-mediated), where the producer is attached to the same messaging engine. In this case, an exception message is generated on each send call. The exception message indicates that the only extant localization has been disabled for send. The producer remains open as usual, and any more send calls succeed if the **Send allowed** property of the localization is reselected (reset to TRUE).
- For a queue point of a mediated destination, if you clear this option then messages from mediation instances are redirected to any available message point. If no message points are available, then the messages are preserved by the bus until a message point becomes available. For any mediation instance (that is, on any server that has a mediation point), if the same server hosts a queue point, and that queue point is the only queue point for the destination, then the mediation changes to the “stopped on error” state.

-receiveAllowed TRUE | FALSE
Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination. For the message point, if you clear this option then any open consumers change state and an exception is generated if the consumer requests a message. Messages can continue to be sent, and accumulate on the message point.

-receiveExclusive TRUE | FALSE
Select this option (setting it to TRUE) to allow only one consumer to attach to a destination. If you select this option, only a single consumer can be attached to each queue point of a queue destination at any one time. Subsequent consumers attempting to attach to a queue point with a consumer already attached are rejected.

-maintainStrictMessageOrder TRUE | FALSE
Select this option (setting it to TRUE) to maintain the order in which a producer sends messages to the destination.

At run time, this property has priority over other configuration property values. For information about the configuration properties that are overridden at run time, see Strict message ordering for bus destinations.

-topicAccessCheckRequired
Include this option if authorization checks are required for access to topics.

-replyDestination
The name of a destination to be appended to any non-empty reverse routing path of messages sent to this destination. This property is intended for use with mediations on reply messages. For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

-replyDestinationBus
The name of the bus on which the reply destination is configured. This property is intended for use with mediations on reply messages. For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

-delegateAuthorizationCheckToTarget
Indicates whether the authorization check is performed on the alias or the target destination. Include this option if you want the authorization check to be performed on the target destination.

-wmqQueueName
To assign the bus destination to a WebSphere MQ queue, specify both the name of the WebSphere MQ server bus member where the destination is assigned (the **-wmqServer** parameter), and the name of the WebSphere MQ queue used to store messages sent to the destination (this parameter). Set the**-wmqServer** parameter to the name you gave when you created the WebSphere MQ server. Set the **-wmqQueueName** parameter to the name allocated to the WebSphere MQ queue by WebSphere MQ administration. Do not specify the **-cluster**, **-node** or **-server** parameters.

-useRFH2 or -mqRfh2Allowed TRUE | FALSE
Determines whether messages sent to the destination have an MQRFH2 header.

When service integration converts a message from the service integration format to WebSphere MQ format, by default it includes an MQRHF2 header in the WebSphere MQ message. This header contains message attributes, such as JMS message attributes, which are not WebSphere MQ message attributes and therefore do not appear in the WebSphere MQ message descriptor (MQMD). Some WebSphere MQ applications cannot process messages that include an MQRFH2 header. If messages sent to this destination will be processed by WebSphere MQ applications that cannot tolerate an MQRFH2, clear this option (setting it to FALSE).

If you are assigning a queue-type destination to a WebSphere MQ server bus member, use the **-useRFH2** parameter. If you are creating an alias destination or a foreign destination, use the **-mqRfh2Allowed** parameter.

-auditAllowed TRUE | FALSE
Clear this option (setting it to FALSE) to prevent the bus from auditing topic level authorization checks when the bus and application server have auditing enabled. The default value is TRUE. You must have Audit Administrator privileges to use this parameter. The parameter is ignored if it is used in the creation of other types of destination.

-defaultForwardRoutingPath
The value to which a message forward routing path is set if the message contains no forward routing path. This identifies a sequential list of intermediary bus destinations that messages must pass through to reach a target bus destination. The format of the field is a list of bus destinations specified as *bus_name*:*destination_name*.

-queuePoints
A list of the queue points used by users of the alias destination. If no queue points are supplied, all queue points can be used. The target destination must be a queue destination in the same bus as the alias destination definition. The target destination must also be a queue destination with multiple queue points.

A queue point is specified in the following form: *destination_name@messaging_engine_name*

-mediationPoints
A list of the mediation points used by users of the alias destination. If no mediation points are supplied, all mediation points can be used. The target destination must be a mediated queue destination in the same bus as the alias destination definition. The target destination must also be a queue destination with multiple mediation points.

A mediation point is specified in the following form: *destination_name@messaging_engine_name*

Example

- Using Jython:

```
wsadmin>AdminTask.createSIBDestination('[-bus bus1 -name myqueue -type QUEUE
-node node1 -server server1]')
'(cells/9994GKCCell01/buses/bus1|sib-destinations.xml#SIBQueue_1098215169998)'
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBDestination {-bus bus1 -name myqueue -type QUEUE
-node node1 -server server1}
(cells/9994GKCCell01/buses/bus1|sib-destinations.xml#SIBQueue_1098215169998)
```

Example: Create a destination alias for "MyDestination1" called "MyAlias1" that can use two queue points:

- Using Jython:

```
wsadmin>AdminTask.createSIBDestination('[-bus bus1 -type ALIAS
-name MyAlias1 -aliasBus bus1 -targetName MyDestination1
-reliability INHERIT -maxReliability INHERIT
-overrideOfQOSByProducerAllowed INHERIT -sendAllowed INHERIT
-receiveAllowed INHERIT
-queuePoints [[-identifier MyDestination1@cluster1.001-bus1]
[-identifier MyDestination1@cluster1.002-bus1]]]')
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBDestination {-bus bus1 -type ALIAS
-name MyAlias1 -aliasBus bus1 -targetName MyDestination1
-reliability INHERIT -maxReliability INHERIT
-overrideOfQOSByProducerAllowed INHERIT -sendAllowed INHERIT
-receiveAllowed INHERIT
-queuePoints {{"MyDestination1@cluster1.001-bus1"}
{"MyDestination1@cluster1.002-bus1"}}}
```

createSIBDestinations command

Use the createSIBDestinations command to create new bus destinations for a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The createSIBDestinations command creates multiple new bus destinations for a service integration bus, all with the same properties. If the destinations created are alias destinations, they all target the same destination. With this command, the new bus destinations can also be localized to a WebSphere MQ server bus member.

Target object

A service integration bus.

Required parameters

-bus

The name of the service integration bus on which to create the bus destinations. To list the names of existing buses, use the listSIBuses command.

-nameList
The list of identifiers by which these destinations are known for administrative purposes.

-type
Indicates the type of bus destination that you want to create:

Alias An alias destination, that provides a level of abstraction between applications and the underlying target bus destinations that hold messages. Applications interact with the alias destination, so the target bus destination can be changed without changing the application.

Foreign
A foreign destination, which identifies a destination on another bus, and enables applications on one bus to access the destination on another bus directly.

Port Represents a particular message and transport binding for an outbound service that communicates with an externally-hosted target web service.

Queue
A queue, for point-to-point messaging.

TopicSpace
A topic space, for publish/subscribe messaging.

WebService
Represents an externally-hosted target web service.

Conditional parameters

None.

Optional parameters

-cluster
To assign the bus destinations to a cluster bus member, specify the name of the cluster. Do not specify the **-node**, **-server** or **-wmqServer** parameters.

-node
To assign the bus destinations to a server bus member, specify both the name of the node on which the server runs and the name of the server. Do not specify the **-cluster** or **-wmqServer** parameters.

-server
To assign the bus destinations to a server bus member, specify both the name of the node on which the server runs and the name of the server. Do not specify the **-cluster** or **-wmqServer** parameters.

-wmqServer
To assign the bus destinations to a WebSphere MQ queue, specify both the name of the WebSphere MQ server bus member where the destination is assigned (this parameter), and the name of the WebSphere MQ queue used to store messages sent to the destinations (the **-wmqQueueName** parameter). Set the**-wmqServer** parameter to the name you gave when you created the WebSphere MQ server. Set the **-wmqQueueName** parameter to the name allocated to the WebSphere MQ queue by WebSphere MQ administration. Do not specify the **-cluster**, **-node** or **-server** parameters.

-aliasBus
If you are creating alias destinations, specify the source bus name of the alias mapping.

-targetBus
If you are creating alias destinations, specify the name of the bus to which the alias destinations are mapped.

-targetName
If you are creating alias destinations, specify the name of the destination to which the alias destinations are mapped.

-foreignBus
If you are creating foreign destinations specify the name of the foreign bus.

-description
Specify a description for the bus destinations for administrative purposes.

-reliability
Specify the default reliability level to assign to messages produced to these destinations when an explicit reliability has not been set by the producer application. Service integration supports five reliability levels (also known as delivery options or qualities of service):

BEST_EFFORT_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

EXPRESS_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

RELIABLE_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails.

RELIABLE_PERSISTENT
Messages might be discarded when a messaging engine fails.

ASSURED_PERSISTENT
Messages are not discarded.

Note: Higher levels of reliability have higher impacts on performance.

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

-maxReliability
Specify the maximum reliability level that is accepted for values specified by producer applications. Service integration supports five reliability levels (also known as delivery options or qualities of service):

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

-nonPersistentReliability

Specify the service integration quality of service to use with nonpersistent WebSphere MQ messages that are received by service integration from a WebSphere MQ network. The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For nonpersistent WebSphere MQ messages received, the default service integration quality of service is RELIABLE_NONPERSISTENT. If you choose to override this default, you will probably choose one

of the other nonpersistent service integration qualities of service BEST_EFFORT_NONPERSISTENT or EXPRESS_NONPERSISTENT. However, you can choose any of the five possible service integration qualities of service:

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

For more information, see ../ae/rjc0014_.dita.

-persistentReliability

Specify the service integration quality of service to use with persistent WebSphere MQ messages that are received by service integration from a WebSphere MQ network. The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For persistent WebSphere MQ messages received, the default service integration quality of service is ASSURED_PERSISTENT. If you choose to override this default, you will probably choose the other persistent service integration quality of service RELIABLE_PERSISTENT. However, you can choose any of the five possible service integration qualities of service:

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

For more information, see ../ae/rjc0014_.dita.

-overrideOfQOSByProducerAllowed TRUE | FALSE

Controls the quality of service for message flows between producers and the destination. Select this option to use the quality of service specified by producers instead of the quality defined for the destination.

-defaultPriority *number*

The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

-maxFailedDeliveries *number*

The maximum number of failed attempts to process a message. After this number of failed attempts, the message is forwarded from the intended destination to its exception destination. Specify a value in the range 0 through 2147483647. A value of 0 (zero) means that if a message cannot be delivered on the first attempt, it is either forwarded to the exception destination or discarded, as defined by the **-exceptionDestination** parameter.

-exceptionDestination *value*

Use these properties to define what happens to any messages that cannot be delivered to this destination.

By default, all messages that cannot be delivered to this destination are rerouted to the system default exception destination for the messaging engine to which this destination is assigned (_SYSTEM.Exception.Destination.*messaging_engine_name*). Use this parameter to override the default value. You can set a specific exception destination for this destination, or you can specify that

undeliverable messages are not rerouted to an exception destination by entering an empty string (""), in which case the maximum failed deliveries count has no effect.

Note: An undeliverable message can block the processing of other messages waiting for delivery to the same destination.

You can use this option and specify no exception destination to preserve message ordering.

-sendAllowed TRUE | FALSE
Clear this option (setting it to FALSE) to stop producers from being able to send messages to these destinations.

- For a queue point of a non-mediated destination, or a mediation point of a mediated destination, if you clear this option then new messages (from attached producers or forwarded from another destination) are redirected to any available message point. If no message points are available, then messages that have already been accepted onto the bus, and new messages from attached producers, are preserved by the bus until a message point becomes available. The only exception to this is the case of a destination with only one message point (queue point or mediation point depending on whether the destination is mediated or non-mediated), where the producer is attached to the same messaging engine. In this case, an exception message is generated on each send call. The exception message indicates that the only extant localization has been disabled for send. The producer remains open as usual, and any more send calls succeed if the **Send allowed** property of the localization is reselected (reset to TRUE).
- For a queue point of a mediated destination, if you clear this option then messages from mediation instances are redirected to any available message point. If no message points are available, then the messages are preserved by the bus until a message point becomes available. For any mediation instance (that is, on any server that has a mediation point), if the same server hosts a queue point, and that queue point is the only queue point for the destination, then the mediation changes to the “stopped on error” state.

-receiveAllowed TRUE | FALSE
Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination. For the message point, if you clear this option then any open consumers change state and an exception is generated if the consumer requests a message. Messages can continue to be sent, and accumulate on the message point.

-receiveExclusive TRUE | FALSE
Select this option (setting it to true) to allow only one consumer to attach to a destination. If you select this option, only a single consumer can be attached to each queue point of a queue destination at any one time. Subsequent consumers attempting to attach to a queue point with a consumer already attached are rejected.

-maintainStrictMessageOrder TRUE | FALSE
Select this option (setting it to TRUE) to maintain the order in which a producer sends messages to a destination.

At run time, this property has priority over other configuration property values. For information about the configuration properties that are overridden at run time, see Strict message ordering for bus destinations.

-topicAccessCheckRequired
Include this option if authorization checks are required for access to topics.

-replyDestination
The name of a destination to be appended to any non-empty reverse routing path of messages sent to this destination. This property is intended for use with mediations on reply messages. For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

-replyDestinationBus
The name of the bus on which the reply destination is configured. This property is intended for use

with mediations on reply messages. For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

-delegateAuthorizationCheckToTarget
Indicates whether the authorization check is performed on the alias or the target destination. Include this option if you want the authorization check to be performed on the target destination.

-wmqQueueName
To assign these bus destinations to a WebSphere MQ queue, specify both the name of the WebSphere MQ server bus member where the destinations are assigned (the **-wmqServer** parameter), and the name of the WebSphere MQ queue used to store messages sent to these destinations (this parameter). Set the**-wmqServer** parameter to the name you gave when you created the WebSphere MQ server. Set the **-wmqQueueName** parameter to the name allocated to the WebSphere MQ queue by WebSphere MQ administration. Do not specify the **-cluster**, **-node** or **-server** parameters.

-useRFH2 or -mqRfh2Allowed TRUE | FALSE
Determines whether messages sent to these destinations have an MQRFH2 header.

When service integration converts a message from the service integration format to WebSphere MQ format, by default it includes an MQRHF2 header in the WebSphere MQ message. This header contains message attributes, such as JMS message attributes, which are not WebSphere MQ message attributes and therefore do not appear in the WebSphere MQ message descriptor (MQMD). Some WebSphere MQ applications cannot process messages that include an MQRFH2 header. If messages sent to this destination will be processed by WebSphere MQ applications that cannot tolerate an MQRFH2, clear this option (setting it to FALSE).

If you are assigning queue-type destinations to a WebSphere MQ server bus member, use the **-useRFH2** parameter. If you are creating alias destinations or foreign destinations, use the **-mqRfh2Allowed** parameter.

-auditAllowed TRUE | FALSE
Clear this option (setting it to FALSE) to prevent the bus from auditing topic level authorization checks when the bus and application server have auditing enabled. The default value is TRUE. You must have Audit Administrator privileges to use this parameter. The parameter is ignored if it is used in the creation of other types of destination.

-defaultForwardRoutingPath
The value to which a message forward routing path is set if the message contains no forward routing path. This identifies a sequential list of intermediary bus destinations that messages must pass through to reach a target bus destination. The format of the field is a list of bus destinations specified as *bus_name*:*destination_name*.

-queuePoints
A list of the queue points used by users of the alias destination. If no queue points are supplied, all queue points can be used. The target destination must be a queue destination in the same bus as the alias destination definition. The target destination must also be a queue destination with multiple queue points.

A queue point is specified in the form *destination_name*@*messaging_engine_name*.

-mediationPoints
A list of the mediation points used by users of the alias destination. If no mediation points are supplied, all mediation points can be used. The target destination must be a mediated queue destination in the same bus as the alias destination definition. The target destination must also be a queue destination with multiple mediation points.

A mediation point is specified in the form *destination_name*@*messaging_engine_name*.

Example

- Using Jython:

```
wsadmin>AdminTask.createSIBDestinations('[-bus bus1 -type QUEUE
-cluster cluster1 -nameList [[-identifier myqueue1][-identifier myqueue2]]]')
'(cells/9994GKCCell01/buses/bus1|sib-destinations.xml#SIBQueue_1098215169998)'
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBDestinations {-bus bus1 -type QUEUE
-cluster cluster1 -nameList {{myqueue1} {myqueue2}}}
(cells/9994GKCCell01/buses/bus1|sib-destinations.xml#SIBQueue_1098215169998)
```

Example: Create aliases for "MyDestination1" called "MyAlias1" and "MyAlias2". These alias destinations give access to a single queue point of the target destination:

- Using Jython:

```
wsadmin>cluster=AdminConfig.list("ServerCluster").splitlines()[0]
wsadmin>AdminTask.createSIBDestinations('[-bus bus1 -type ALIAS
-nameList [[-identifier MyAlias1][-identifier MyAlias2]] -aliasBus bus1
-targetName MyDestination1 -reliability INHERIT -maxReliability INHERIT
-overrideOfQOSByProducerAllowed INHERIT -sendAllowed INHERIT
-receiveAllowed INHERIT -queuePoints [[-identifier MyDestination1@cluster1.001-bus1]
[-identifier MyDestination1@cluster1.002-bus1]]]')
```

- Using Jacl:

```
wsadmin>set cluster [ lindex [ $AdminConfig list ServerCluster ] 1 ]
wsadmin>$AdminTask createSIBDestinations {-bus bus1 -type ALIAS
-nameList {{MyAlias1} {MyAlias2}} -aliasBus bus1
-targetName MyDestination1 -reliability INHERIT -maxReliability INHERIT
-overrideOfQOSByProducerAllowed INHERIT -sendAllowed INHERIT
-receiveAllowed INHERIT -queuePoints {{"MyDestination1@cluster1.001-bus1"}
{"MyDestination1@cluster1.002-bus1}}}
```

deleteSIBDestination command

Use the deleteSIBDestination command to delete a bus destination.

This command deletes the named destination on the named bus, and deletes all related message points.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The deleteSIBDestination command deletes the named destination on the named bus, and deletes all related message points. This command also finds and cleans up all destinations that are assigned to WebSphere MQ server bus members, and all destinations that are mediated to a WebSphere MQ queue.

This command does not delete messages from WebSphere MQ queues, and it does not delete the queues.

Target object

A bus destination.

Required parameters

-bus
The name of the service integration bus on which the bus destination is configured. You can use the listSIBuses command to list the names of existing buses.

-name
The identifier by which this destination is known for administrative purposes.

-aliasBus
If the destination to be deleted is an alias destination and was created with the **aliasBus** parameter, then the same value must be used to delete the destination.

-foreignBus
If the destination to be deleted is a foreign destination, then the foreign destination parameter must be supplied.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminTask.deleteSIBDestination("-bus abus -name myqueue")

wsadmin>AdminConfig.save()
```

- Using Jacl:

```
wsadmin>$AdminTask deleteSIBDestination {-bus abus -name myqueue}

wsadmin>$AdminConfig save
```

deleteSIBDestinations command

Use the deleteSIBDestinations command to delete bus destinations.

This command deletes the named destinations on the named bus, and deletes all related message points.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when it is used with WebSphere Application Server Version 6.1.0 (Fix Pack 15) or later. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The deleteSIBDestinations command deletes the named destinations on the named bus, and deletes all related message points. This command also finds and cleans up all destinations that are assigned to WebSphere MQ server bus members, and all destinations that are mediated to a WebSphere MQ queue. This command does not delete messages from WebSphere MQ queues, and it does not delete the queues.

Target object

Bus destinations.

Required parameters

-bus
The name of the service integration bus on which the bus destination is configured. You can use the listSIBuses command to list the names of existing buses.

-nameList
The identifiers by which these destinations are known for administrative purposes.

-aliasBus
If the destination to be deleted is an alias destination and was created with the **aliasBus** parameter, then the same value must be used to delete the destination.

-foreignBus
If the destination to be deleted is a foreign destination, then the foreign destination parameter must be supplied.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminTask.deleteSIBDestinations(["-bus" , "abus" , "-nameList" , [["myqueue1"] ,["myqueue2"]]])

wsadmin>$AdminConfig save
```

- Using Jacl:

```
wsadmin>$AdminTask deleteSIBDestinations {-bus myBus -nameList {{myqueue1} {myqueue2}} }

wsadmin>$AdminConfig save
```

listSIBDestinations command

Use the listSIBDestinations command to list the bus destinations for a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all the bus destinations on a bus.

Target object

A bus.

Required parameters

-bus *busname*
The name of the service integration bus on which the bus destinations are configured. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

-type Queue | TopicSpace | WebService | Port
To list destinations of the specified type.

Example

- Using Jython:

```
wsadmin>AdminTask.listSIBDestinations("-bus abus")
'(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBTopicSpace_1098181446388)
(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBQueue_1098181503600)
(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBQueue_1098184221748)'

AdminTask.listSIBDestinations("-bus abus -type TopicSpace")
'(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBTopicSpace_1098181446388)'
```

- Using Jacl:

```
wsadmin>$AdminTask listSIBDestinations {-bus abus}
(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBTopicSpace_1098181446388)
(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBQueue_1098181503600)
(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBQueue_1098184221748)

wsadmin>$AdminTask listSIBDestinations {-bus abus -type TopicSpace}
(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBTopicSpace_1098181446388)
```

mediateSIBDestination command

Use the mediateSIBDestination command to mediate a bus destination for a service integration bus.

Mediating a destination associates a mediation with a selected bus destination. At run time, the mediation applies some message processing to the messages handled by the bus destination. Note that you can only mediate a destination with a single mediation at a time. You can mediate more than one destination with the same mediation.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command mediates a bus destination for a service integration bus. There are parameters that you can use to create a mediation queue point on a WebSphere MQ server, and create the corresponding mediation execution point on a cluster or a server.

You should only use the **node**, **server**, and **cluster** parameters in WebSphere Application Server environments that support server clusters.

Note: If you are mediating a destination where the mediation point is a queue on a WebSphere MQ server (using the **wmqServer** parameter), you must specify where the mediation code runs. If the mediation code runs in a service integration bus member, you must specify a mediation execution point by using the **node**, **server**, and **cluster** parameters. If the mediation code runs externally, you must omit the **node**, **server**, and **cluster** parameters.

- If you are not using a mediation point that is a queue on a WebSphere MQ server, use the **node**, **server**, and **cluster** parameters to specify where the mediation point is located. This is also where the mediation runs. If you omit these parameters, the mediation point defaults to the bus member to which the destination being mediated is assigned.
- If you are using a mediation point that is a queue on a WebSphere MQ server, use the **wqmServer** and **wmqQueueName** parameters to specify the mediation point.
  - If you omit the **node**, **server**, and **cluster** parameters, the service integration bus assumes that the mediation process is performed by an external WebSphere MQ application.
  - If you want to assign the mediation point to a WebSphere MQ server bus member, use the **wmqServer** and **wmqQueueName** parameters to specify a WebSphere MQ queue. In this situation, because you have omitted the **node**, **server**, and **cluster** parameters, an external WebSphere MQ application can run the mediation. The **node**, **server**, and **cluster** parameters are used for nominating the bus member where the mediation code runs.

Target object

None.

Required parameters

-bus *bus_name* **-destinationName** *destination_name*
The destination to be mediated.

-mediationName *mediation_name*
The name of the mediation to be applied to the bus destination. This mediation must exist before this command can be used.

Conditional parameters

-wmqServer *mq_server_name* **-wmqQueueName** *mq_queue_name*
[Queue or web service destination] To assign the mediation point to a WebSphere MQ queue, specify both the name of the WebSphere MQ server bus member where the mediation point is to be assigned, and the name of the WebSphere MQ queue to be used to store messages ready for mediation. *mq_server_name* is the name of the WebSphere MQ server as specified in the **-name** parameter when creating the WebSphere MQ server. *mq_queue_name* is the name allocated to the WebSphere MQ queue by WebSphere MQ administration.

-node *node_name* **-server** *server_name*
[Not topic space] To mediate the bus destination to a server bus member, specify both the name of the node on which the server runs and the name of the server.

-cluster *cluster_name*
[Not topic space] To mediate the bus destination to a cluster bus member, specify the name of the cluster.

This option should be used only in WebSphere Application Server environments that support server clusters.

Optional parameters

-nonPersistentReliability

Specify the service integration quality of service to use with nonpersistent WebSphere MQ messages that are received by service integration from a WebSphere MQ network. The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For nonpersistent WebSphere MQ messages received, the default service integration quality of service is RELIABLE_NONPERSISTENT. If you choose to override this default, you will probably choose one of the other nonpersistent service integration qualities of service BEST_EFFORT_NONPERSISTENT or EXPRESS_NONPERSISTENT. However, you can choose any of the five possible service integration qualities of service:

BEST_EFFORT_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

EXPRESS_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

RELIABLE_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails.

RELIABLE_PERSISTENT
Messages might be discarded when a messaging engine fails.

ASSURED_PERSISTENT
Messages are not discarded.

Note: Higher levels of reliability have higher impacts on performance.

For more information, see ../ae/rjc0014_.dita.

-persistentReliability

Specify the service integration quality of service to use with persistent WebSphere MQ messages that are received by service integration from a WebSphere MQ network. The messages in a WebSphere MQ network have their own quality of service level. This is either persistent or non-persistent. When these messages are received by a service integration application, they are assigned a service integration quality of service level that depends on their WebSphere MQ quality of service level.

For persistent WebSphere MQ messages received, the default service integration quality of service is ASSURED_PERSISTENT. If you choose to override this default, you will probably choose the other persistent service integration quality of service RELIABLE_PERSISTENT. However, you can choose any of the five possible service integration qualities of service:

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

For more information, see ../ae/rjc0014_.dita.

-useRFH2
Determines whether service integration technologies includes an RFH2 header in messages it places on the mediation point. Possible values are:

TRUE

FALSE

The default value is TRUE.

-maintainStrictMessageOrder
Maintain strict message order. Possible values are:

Selected
Maintains the order in which a producer sends messages to the destination.

At run time, this property has priority over other configuration property values. For information on the configuration properties that are overridden at run time, see Strict message ordering for bus destinations.

Cleared
Message order is not preserved for this destination.

Example

- Using Jython:

```
wsadmin>AdminTask.mediateSIBDestination("-bus abus -destinationName myqueue
 -mediationName filterMed -cluster cluster1")
```

- Using Jacl:

```
wsadmin>$AdminTask mediateSIBDestination {-bus abus -destinationName myqueue
 -mediationName filterMed -cluster cluster1}
```

modifySIBDestination command

Use the modifySIBDestination command to change properties of a bus destination for a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The modifySIBDestination changes properties of a bus destination for a service integration bus.

Target object

A bus destination.

Required parameters

-bus
The name of the service integration bus on which the bus destination is configured. You can use the listSIBuses command to list the names of existing buses.

-name
The identifier by which this destination is known for administrative purposes.

Conditional parameters

None.

Optional parameters

-description
Specify a description for the bus destination, for administrative purposes.

-reliability
Specify the default reliability level to assign to a message produced to this destination when an explicit reliability has not been set by the producer application. Service integration supports five reliability levels (also known as delivery options or qualities of service):

BEST_EFFORT_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

EXPRESS_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

RELIABLE_NONPERSISTENT
Messages are discarded when a messaging engine stops or fails.

RELIABLE_PERSISTENT
Messages might be discarded when a messaging engine fails.

ASSURED_PERSISTENT
Messages are not discarded.

Note: Higher levels of reliability have higher impacts on performance.

For more information about service integration reliability levels, see Message reliability levels - JMS delivery mode and service integration quality of service.

-overrideOfQOSByProducerAllowed TRUE | FALSE
Controls the quality of service for message flows between producers and the destination. Select this option to use the quality of service specified by producers instead of the quality defined for the destination.

-defaultPriority *number*
The default priority assigned to messages sent to this destination when a priority has not been set by the producer.

-maxFailedDeliveries *number*
The maximum number of failed attempts to process a message. After this number of failed attempts, the message is forwarded from the intended destination to its exception destination. Specify a value in the range 0 through 2147483647. A value of 0 (zero) means that if a message cannot be delivered on the first attempt, it is either forwarded to the exception destination or discarded, as defined by the **-exceptionDestination** parameter.

-exceptionDestination *value*
Use these properties to define what happens to any messages that cannot be delivered to this destination.

By default, all messages that cannot be delivered to this destination are rerouted to the system default exception destination for the messaging engine to which this destination is assigned (_SYSTEM.Exception.Destination.*messaging_engine_name*). Use this parameter to override the default value. You can set a specific exception destination for this destination, or you can specify that undeliverable messages are not rerouted to an exception destination by entering an empty string (""), in which case the maximum failed deliveries count has no effect.

Note: An undeliverable message can block the processing of other messages waiting for delivery to the same destination.

You can use this option and specify no exception destination to preserve message ordering.

-sendAllowed TRUE | FALSE
Clear this option (setting it to FALSE) to stop producers from being able to send messages to this destination.

- For a queue point of a non-mediated destination, or a mediation point of a mediated destination, if you clear this option then new messages (from attached producers or forwarded from another destination) are redirected to any available message point. If no message points are available, then messages that have already been accepted onto the bus, and new messages from attached

producers, are preserved by the bus until a message point becomes available. The only exception to this is the case of a destination with only one message point (queue point or mediation point depending on whether the destination is mediated or non-mediated), where the producer is attached to the same messaging engine. In this case, an exception message is generated on each send call. The exception message indicates that the only extant localization has been disabled for send. The producer remains open as usual, and any more send calls succeed if the **Send allowed** property of the localization is reselected (reset to TRUE).

- For a queue point of a mediated destination, if you clear this option then messages from mediation instances are redirected to any available message point. If no message points are available, then the messages are preserved by the bus until a message point becomes available. For any mediation instance (that is, on any server that has a mediation point), if the same server hosts a queue point, and that queue point is the only queue point for the destination, then the mediation changes to the “stopped on error” state.

-receiveAllowed TRUE | FALSE
Clear this option (setting it to false) to prevent consumers from being able to receive messages from this destination. For the message point, if you clear this option then any open consumers change state and an exception is generated if the consumer requests a message. Messages can continue to be sent, and accumulate on the message point.

-receiveExclusive TRUE | FALSE
Select this option (setting it to true) to allow only one consumer to attach to a destination. If you select this option, only a single consumer can be attached to each queue point of a queue destination at any one time. Subsequent consumers attempting to attach to a queue point with a consumer already attached are rejected.

-maintainStrictMessageOrder TRUE | FALSE
Select this option (setting it to TRUE) to maintain the order in which a producer sends messages to the destination.

At run time, this property has priority over other configuration property values. For information about the configuration properties that are overridden at run time, see Strict message ordering for bus destinations.

-topicAccessCheckRequired
Include this option if authorization checks are required for access to topics.

-replyDestination
The name of a destination to be appended to any non-empty reverse routing path of messages sent to this destination. This property is intended for use with mediations on reply messages. For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

-replyDestinationBus
The name of the bus on which the reply destination is configured. This property is intended for use with mediations on reply messages. For more information about the use of this property, see “Configuring a destination reverse routing path” on page 1989.

-delegateAuthorizationCheckToTarget
Indicates whether the authorization check is performed on the alias or the target destination. Include this option if you want the authorization check to be performed on the target destination.

-auditAllowed TRUE | FALSE
Clear this option (setting it to FALSE) to prevent the bus from auditing topic level authorization checks when the bus and application server have auditing enabled. The default value is TRUE. You must have Audit Administrator privileges to use this parameter. The parameter is ignored if it is used in the creation of other types of destination.

-defaultForwardRoutingPath
The value to which a message forward routing path is set if the message contains no forward routing

path. This identifies a sequential list of intermediary bus destinations that messages must pass through to reach a target bus destination. The format of the field is a list of bus destinations specified as *bus_name*:*destination_name*.

-queuePoints
A list of the queue points used by users of the alias destination. If no queue points are supplied, all queue points can be used. The target destination must be a queue destination in the same bus as the alias destination definition. The target destination must also be a queue destination with multiple queue points.

A queue point is specified in the following form: *destination_name@ messaging_engine_name*

-useAllQueuePoints TRUE | FALSE
If you set this option to TRUE all available queue points are used whereas, if you set this option to FALSE, only those queue points in the list specified by the **-queuePoints** option are used.

-mediationPoints
A list of the mediation points used by users of the alias destination. If no mediation points are supplied, all mediation points can be used. The target destination must be a mediated queue destination in the same bus as the alias destination definition. The target destination must also be a queue destination with multiple mediation points.

A mediation point is specified in the following form: *destination_name@ messaging_engine_name*

-useAllMediationPoints TRUE | FALSE
If you set this option to TRUE all available queue points are used whereas, if you set this option to FALSE, only those queue points in the list specified by the **-mediationPoints** option are used.

Example

- Using Jython:

```
wsadmin>AdminTask.showSIBDestination(["-bus", "abus", "-name", "myqueue"])
'{receiveExclusive=false, defaultForwardRoutingPath=[], defaultPriority=0,
exceptionDestination=_SYSTEM.Exception.Destination.node01.aserver-abus,
uuid=97CC75AC71E5932CAB3417AC,  overrideOfQOSByProducerAllowed=true,
sendAllowed=true, maxFailedDeliveries=5,
maxReliability=ASSURED_PERSISTENT, reliability=ASSURED_PERSISTENT,
receiveAllowed=true, identifier=myqueue}'

wsadmin>AdminTask.modifySIBDestination(["-bus", "abus", "-name", "myqueue",
 "-receiveAllowed", "FALSE"])

wsadmin>AdminTask.showSIBDestination(["-bus", "abus", "-name", "myqueue"])
'{receiveExclusive=false, defaultForwardRoutingPath=[], defaultPriority=0,
exceptionDestination=_SYSTEM.Exception.Destination.node01.aserver-abus,
uuid=97CC75AC71E5932CAB3417AC, overrideOfQOSByProducerAllowed=true,
sendAllowed=true, maxFailedDeliveries=5,
maxReliability=ASSURED_PERSISTENT, reliability=ASSURED_PERSISTENT,
receiveAllowed=false, identifier=myqueue}'
```

- Using Jacl:

```
wsadmin>$AdminTask showSIBDestination {-bus abus -name myqueue}
{receiveExclusive=false, defaultForwardRoutingPath=[], defaultPriority=0,
exceptionDestination=_SYSTEM.Exception.Destination.node01.aserver-abus,
uuid=97CC75AC71E5932CAB3417AC, overrideOfQOSByProducerAllowed=true,
sendAllowed=true, maxFailedDeliveries=5,
maxReliability=ASSURED_PERSISTENT, reliability=ASSURED_PERSISTENT,
receiveAllowed=true, identifier=myqueue}

wsadmin>$AdminTask modifySIBDestination {-bus abus -name myqueue
-receiveAllowed FALSE}
(cells/9994GKCCell01/buses/abus|sib-destinations.xml#SIBQueue_1098215169998)

wsadmin>$AdminTask showSIBDestination {-bus abus -name myqueue}
{receiveExclusive=false, defaultForwardRoutingPath=[], defaultPriority=0,
```

```
exceptionDestination=_SYSTEM.Exception.Destination.node01.aserver-abus,
uuid=97CC75AC71E5932CAB3417AC, overrideOfQOSByProducerAllowed=true,
sendAllowed=true, maxFailedDeliveries=5,
maxReliability=ASSURED_PERSISTENT, reliability=ASSURED_PERSISTENT,
receiveAllowed=false, identifier=myqueue}
```

Example: Modify a destination alias "MyAlias2" to use a subset of the available queue points and mediation points:

- Using Jython:

```
cluster=AdminConfig.list("ServerCluster").splitlines()[0]
Qp1=AdminConfig.list("SIBQueueLocalizationPoint" , cluster).splitlines()[0]
Mp1=AdminConfig.list("SIBMediationLocalizationPoint").splitlines()[0]
AdminTask.modifySIBDestination(["-bus" , "bus1" , "-name" , "MyAlias2",
 "-queuePoints", [[Qp1]], "-mediationPoints" , [[Mp1]]])
```

- Using Jacl:

```
set cluster [ lindex [ $AdminConfig list ServerCluster ] 1 ]
set Qp1 [ lindex [ $AdminConfig list SIBQueueLocalizationPoint $cluster ] 0 ]
set Mp1 [ lindex [ $AdminConfig list SIBMediationLocalizationPoint ] 0 ]
$AdminTask modifySIBDestination {-bus bus1 -name MyAlias2
 -queuePoints [[$Qp1]]-mediationPoints [[$Mp1]]}
```

Example: Modify a destination alias to remove any limitation on the queue points used:

- Using Jython:

```
AdminTask.modifySIBDestination(["-bus" , "bus1" , "-name" , "MyAlias2",
 "-queuePoints", [[]]])
```

- Using Jacl:

```
$AdminTask modifySIBDestination {-bus bus1 -name MyAlias2
-useAllQueuePoints=true -useAllMediationPoints=true}
```

showSIBDestination command

Use the showSIBDestination command to list the property values for a bus destination.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists details about properties of a bus destination.

Target object

A bus destination.

Required parameters

-bus *busname*
The name of the service integration bus on which the bus destination is configured. You can use the listSIBuses command to list the names of existing buses.

-name *destname*
The identifier by which this destination is known for administrative purposes.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminTask.showSIBDestination("-bus abus -name myqueue")
'{receiveExclusive=false, defaultForwardRoutingPath=[], defaultPriority=0,
 exceptionDestination=_SYSTEM.Exception.Destination.node01.server1-abus,
 uuid=97CC75AC71E5932CAB3417AC, overrideOfQOSByProducerAllowed=true,
 sendAllowed=true, maxFailedDeliveries=5,
 maxReliability=ASSURED_PERSISTENT, reliability=ASSURED_PERSISTENT,
 receiveAllowed=true, identifier=myqueue}'
```

- Using Jacl:

```
wsadmin>$AdminTask showSIBDestination {-bus abus -name myqueue}
{receiveExclusive=false, defaultForwardRoutingPath=[], defaultPriority=0,
 exceptionDestination=_SYSTEM.Exception.Destination.node01.server1-abus,
 uuid=97CC75AC71E5932CAB3417AC, overrideOfQOSByProducerAllowed=true,
 sendAllowed=true, maxFailedDeliveries=5,
 maxReliability=ASSURED_PERSISTENT, reliability=ASSURED_PERSISTENT,
 receiveAllowed=true, identifier=myqueue}
```

unmediateSIBDestination command

Use the unmediateSIBDestination command to remove a mediation from a service integration bus destination.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes a mediation from a bus destination for a service integration bus. The mediation is left on the bus, in case it is needed to mediate other bus destinations.

If you unmediate a destination that is assigned to a WebSphere MQ server bus member, or a destination that has a mediation point on a WebSphere MQ server bus member, the mediation process stops, and messages already queued on the mediation point are not mediated.

Target object

None.

Required parameters

-bus *busname*
The name of the service integration bus on which the bus destination is to be created. You can use the listSIBuses command to list the names of existing buses.

-destinationName *destname*
The identifier by which this destination is known for administrative purposes.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminTask.listSIBMediations(["-bus", "abus"])
'(cells/9994GKCCell01/buses/abus|sib-mediations.xml#
SIBDestinationMediation_1098217858584)'

wsadmin>AdminTask.unmediateSIBDestination(["-bus", "abus", "-destinationName", "myqueue"])
```

- Using Jacl:

```
wsadmin>$AdminTask listSIBMediations {-bus abus}
(cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098217858584)

wsadmin>$AdminTask unmediateSIBDestination {-bus abus -destinationName myqueue}
```

SIBAdminCommands: Mediation administrative commands for the AdminTask object

You can use these administrative commands to manage mediations.

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and JACL.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

createSIBMediation command

Use the createSIBMediation command to create a new mediation.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a new mediation.

Target object

None.

Required parameters

-bus *busname*
The name of the service integration bus on which the mediation is to be created. You can use the listSIBuses command to list the names of existing buses.

-mediationName *mediation_name*
The name by which this mediation is known for administrative purposes.

-handlerListName
The name of the handler list that was defined when the mediation was deployed.

Conditional parameters

None.

Optional parameters

-description *text*
An optional description for the mediation, for administrative purposes.

-globalTransaction TRUE | FALSE
Whether or not a global transaction is started for each message processed.

FALSE
A local transaction is started for each message processed. You only have to select this option for mediations that access other resource managers such as databases, or interact with enterprise beans that require a global transaction.

TRUE A global transaction is started for each message processed.

-allowConcurrentMediation TRUE | FALSE
Select this option (setting it to true) to apply the mediation to multiple messages concurrently. Message ordering is not preserved. The default option is false.

TRUE Apply the mediation to multiple messages concurrently, and preserve message ordering.

FALSE
Apply the mediation to a single message at a time. This setting is required to ensure that message ordering is preserved.

-selector *text*
Controls which messages are sent to the mediation. If a message matches the rule defined by the selector text string, then the mediation is applied to the message.

If the message does not match the rule defined by the selector text string, then the message is not mediated. If a message contains both Selector and Discriminator, it must match both rules for the message to be mediated. If either the Selector or the Discriminator rule does not match, the message is not mediated.

-discriminator *text*
Discriminator

Compare this property with the selector property. The rule specified by the selector examines the header and properties of the message, whereas the discriminator examines the topic of the message. If a message contains both selector and discriminator, it must match both rules for the message to be mediated. If either the selector or the discriminator rule does not match, the message is not mediated.

Example

- Using Jython:

```
wsadmin>AdminTask.createSIBMediation("-bus abus -mediationName switchMed
 -handlerListName switchHandler")
'(cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098219493014)'
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBMediation {-bus abus -mediationName switchMed
 -handlerListName switchHandler}
(cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098219493014)
```

deleteSIBMediation command

Use the deleteSIBMediation command to delete a mediation.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes a mediation. If the mediation point relates to a WebSphere MQ queue, the contents of the mediation point are not deleted.

Target object

None.

Required parameters

-bus *busname*
The name of the service integration bus on which the mediation is to be created. You can use the listSIBuses command to list the names of existing buses.

-mediationName *mediation_name*
The name by which this mediation is known for administrative purposes.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

  ```
  wsadmin>AdminTask.deleteSIBMediation(["-bus", "abus", "-mediationName", "switchMed"])
  ```

- Using Jacl:

  ```
  wsadmin>$AdminTask deleteSIBMediation {-bus abus -mediationName switchMed}
  ```

listSIBMediations command

Use the listSIBMediations command to list mediations on a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

This command lists all mediations on a service integration bus.

Target object

A service integration bus

Required parameters

-bus *busname*
The name of the service integration bus on which the mediations are configured. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
wsadmin>AdminTask.listSIBMediations("-bus abus")
(cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098217858584)
(cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098220051588)
```

- Using Jacl:

```
wsadmin>$AdminTask listSIBMediations {-bus abus}
(cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098217858584)
(cells/9994GKCCell01/buses/abus|sib-mediations.xml#SIBDestinationMediation_1098220051588)
```

modifySIBMediation command

Use the modifySIBMediation command to change the properties of a mediation.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command changes properties of a mediation.

Target object

A mediation.

Required parameters

-bus *busname*
The name of the service integration bus on which the mediation is to be created. You can use the listSIBuses command to list the names of existing buses.

-mediationName *mediation_name*
The name by which this mediation is known for administrative purposes.

Conditional parameters

None.

Optional parameters

-description *text*
An optional description for the mediation, for administrative purposes.

-handlerListName
The name of the handler list that was defined when the mediation was deployed.

-globalTransaction TRUE | FALSE
Whether or not a global transaction is started for each message processed.

FALSE
A local transaction is started for each message processed. You only have to select this option for mediations that access other resource managers such as databases, or interact with enterprise beans that require a global transaction.

TRUE A global transaction is started for each message processed.

-allowConcurrentMediation TRUE | FALSE
Select this option (setting it to true) to apply the mediation to multiple messages concurrently. Message ordering is not preserved. The default option is false.

TRUE Apply the mediation to multiple messages concurrently, and preserve message ordering.

FALSE
Apply the mediation to a single message at a time. This setting is required to ensure that message ordering is preserved.

-selector *text*
Controls which messages are sent to the mediation. If a message matches the rule defined by the selector text string, then the mediation is applied to the message.

If the message does not match the rule defined by the selector text string, then the message is not mediated. If a message contains both Selector and Discriminator, it must match both rules for the message to be mediated. If either the Selector or the Discriminator rule does not match, the message is not mediated.

-discriminator *text*

Discriminator

Compare this property with the selector property. The rule specified by the selector examines the header and properties of the message, whereas the discriminator examines the topic of the message. If a message contains both selector and discriminator, it must match both rules for the message to be mediated. If either the selector or the discriminator rule does not match, the message is not mediated.

Example

- Using Jython:

```
wsadmin>AdminTask.showSIBMediation("-bus abus -mediationName switchMed")
{uuid 39588C4821BB046E}
{selector {}}
{contextInfo {}}
{discriminator {}}
{allowConcurrentMediation false}
{globalTransaction false}
{mediationName switchMed}
{handlerListName switchHandler}
{description {}}

wsadmin>AdminTask.modifySIBMediation(["-bus", "abus",
 "-mediationName", "switchMed",
 "-selector", ["JMSXDeliveryCount > 1000"]] )

wsadmin>AdminTask.showSIBMediation("-bus abus
-mediationName switchMed")
{uuid 39588C4821BB046E}
{selector {JMSXDeliveryCount > 1000}}
{contextInfo {}}
{discriminator {}}
{allowConcurrentMediation false}
{globalTransaction false}
{mediationName switchMed}
{handlerListName switchHandler}
{description {}}
```

- Using Jacl:

```
wsadmin>$AdminTask showSIBMediation {-bus abus -mediationName switchMed}
{uuid 39588C4821BB046E}
{selector {}}
{contextInfo {}}
{discriminator {}}
{allowConcurrentMediation false}
{globalTransaction false}
{mediationName switchMed}
{handlerListName switchHandler}
{description {}}

wsadmin>$AdminTask modifySIBMediation {-bus abus -mediationName switchMed
 -selector {JMSXDeliveryCount > 1000}}

wsadmin>$AdminTask showSIBMediation {-bus abus -mediationName switchMed}
{uuid 39588C4821BB046E}
{selector {JMSXDeliveryCount > 1000}}
{contextInfo {}}
{discriminator {}}
{allowConcurrentMediation false}
{globalTransaction false}
{mediationName switchMed}
{handlerListName switchHandler}
{description {}}
```

showSIBMediation command

Use the showSIBMediation command to list the property values for a mediation.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command changes properties of a mediation.

Target object

A mediation.

Required parameters

-bus *busname*
The name of the service integration bus on which the mediation is to be created. You can use the listSIBuses command to list the names of existing buses.

-mediationName *mediation_name*
The name by which this mediation is known for administrative purposes.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

  ```
  wsadmin>AdminTask.showSIBMediation("-bus abus -mediationName switchMed")
  {uuid 39588C4821BB046E}
  {selector {JMSXDeliveryCount > 1000}}
  {contextInfo {}}
  {discriminator {}}
  {allowConcurrentMediation false}
  {globalTransaction false}
  {mediationName switchMed}
  {handlerListName switchHandler}
  {description {}}
  ```

- Using Jacl:

  ```
  wsadmin>$AdminTask showSIBMediation {-bus abus -mediationName switchMed}
  {uuid 39588C4821BB046E}
  {selector {JMSXDeliveryCount > 1000}}
  ```

```
{contextInfo {}}
{discriminator {}}
{allowConcurrentMediation false}
{globalTransaction false}
{mediationName switchMed}
{handlerListName switchHandler}
{description {}}
```

SIBAdminCommands: WebSphere MQ server administrative commands for the AdminTask object

Use command scripts to create, modify, list, show and delete WebSphere MQ servers and server bus members. These commands do not support the automatic resource discovery feature provided by the administrative console.

Decide which method to use to configure these resources. You can configure WebSphere MQ server resources by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Using a WebSphere MQ server to integrate WebSphere MQ queues into a bus” on page 573.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Note: To add a WebSphere MQ server as a bus member, use the addSIBusMember command.

createSIBWMQServer command

Use the createSIBWMQServer command to create a new WebSphere MQ server at cell scope.

You can create a new WebSphere MQ server by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Creating a WebSphere MQ server definition” on page 574.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command creates a new WebSphere MQ server at cell scope, and uses the supplied values to populate its parameters.

Note: When creating a WebSphere MQ server, it is important to understand the significance of the **-name** and **-serverName** parameters. For example, if WebSphere Application Server administration created a WebSphere MQ server object with the name “My European area server” that represented a WebSphere MQ queue manager with the **serverName** QM1. WebSphere Application Server administration could then create a second WebSphere MQ server object with the name “My UK country server”, that also represented the same WebSphere MQ queue manager with the **serverName** QM1.

Target object

A WebSphere Application Server cell.

Required parameters

-name
The name of the WebSphere MQ server. This value is for administrative purposes only and can be decided by the administrator. The name is only meaningful inside WebSphere Application Server administration, and must be unique at cell level. There is no default value. This parameter cannot be modified.

-serverName
The name of the queue manager or queue-sharing group. This value is the name by which the queue manager or queue-sharing group is identified, and is allocated by WebSphere MQ administration to that WebSphere MQ object. The WebSphere Application Server administrator must always use the name allocated by WebSphere MQ administration.

-host
The host to which a connection is established for communicating with a queue manager or queue-sharing group. This value is the host name or the IP address of the queue manager or queue-sharing group, that this WebSphere MQ server represents. The value is a string and must be one of the following:

- symbolic host name
- IPv4 address
- IPv6 address

-transportChain
The channel framework outbound transport chain to use when establishing a connection with WebSphere MQ. If you do not specify this option, a default value of OutboundBasicWMQClient is assumed.

Conditional parameters

None

Optional parameters

The optional host, port, channel, and authentication alias attributes together specify the connection access path to this WebSphere MQ server, for messaging applications running in service integration. For more information, see WebSphere MQ server: Connection and authentication.

-port
The TCP/IP port number on which the queue manager or queue-sharing group that this WebSphere MQ server represents listens. The default value is 1414.

-channel
The WebSphere MQ client channel name to use when connecting to the queue manager or queue-sharing group that this WebSphere MQ server represents. This value is the name allocated by WebSphere MQ administration to the WebSphere MQ object, and must always be used by WebSphere Application Server administration. The default value is SYSTEM.DEF.SVRCONN.

-description
A short description of the WebSphere MQ server. This value is used for administrative purposes only.

-securityAuthAlias
The authentication alias to use when connecting to a queue manager or queue-sharing group. This parameter should not be confused with the discovery authentication alias.

-trustUserIds
Determines whether user IDs received in messages from WebSphere MQ are passed on with the messages by the service integration bus. The application user ID is always set from the **jsAppUserId** RFH2 value. If this is not present (either because the key/value pair is not present in the RFH2 header, or because the message does not have a RFH2 header), this field is not set. If you set this value to FALSE, the user ID is overwritten with the WebSphere MQ server name. This parameter has two possible values:

TRUE User IDs are propagated into messages.

FALSE
User IDs are not propagated into messages.

The default is TRUE.

-allowDiscovery
Determines whether automated discovery of WebSphere MQ resources is performed. This parameter has two possible values:

TRUE Automated discovery is used.

FALSE
Automated discovery is not used.

The default is TRUE.

-discoveryAuthAlias
The authentication alias to use when establishing a resource discovery connection to the queue manager or queue-sharing group. This value should not be confused with the security authentication alias.

-replyToQueue
The reply-to queue to use for resource discovery. This value is the name allocated by WebSphere MQ administration to the WebSphere MQ object, and must be the name of a model queue for a temporary dynamic queue. The WebSphere Application Server administrator must always use the name allocated by WebSphere MQ administration. The default is SYSTEM.DEFAULT.MODEL.QUEUE.

-type
Determines whether the WebSphere MQ server object is either a queue manager or a queue-sharing group, as determined by WebSphere Application Server administration. This parameter had two possible values:

MQ_QUEUE_MANAGER
The WebSphere MQ server represents a queue manager. If you select this value, the resource discovery process retrieves queue names that belong to queue managers.

MQ_QUEUE_SHARING_GROUP
The WebSphere MQ server represents a queue-sharing group. If you select this value, the resource discovery process retrieves queue names that belong to queue-sharing groups.

-bindingsMode
Determines whether bindings transport mode connections are used when connecting to a queue manager or queue-sharing group. Bindings mode connection is available if the application server and the queue manager are on the same node. It is only possible to connect to a single queue manager in bindings mode, even if multiple queue managers exist on the same node. This parameter has two possible values:

TRUE Bindings mode is used if available. If you select this option and bindings mode is not available, the connections mechanism defaults to client transport mode.

FALSE
Client mode is always used.

Example

- Using Jython:

```
wsadmin>AdminTask.createSIBWMQServer(["-name", "Finance dept QM",
 "-serverName", "FDQM", "-type", "MQ_QUEUE_MANAGER", "-bindingsMode", "true",
 "-host", "findep01.ibm.com", "-port", 1414,
 "-transportChain", "OutboundSecureWMQClient"])
```

- Using Jacl:

```
wsadmin>$AdminTask createSIBWMQServer {-name "Finance dept QM"
-serverName FDQM -type MQ_QUEUE_MANAGER -bindingsMode true
-host findep01.ibm.com -port 1414
-transportChain OutboundSecureWMQClient}
```

modifySIBWMQServer command

Use the modifySIBWMQServer command to modify a WebSphere MQ server.

You can modify a WebSphere MQ server by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Modifying a WebSphere MQ server definition” on page 575.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command modifies a WebSphere MQ server. It is not possible to modify the name or type attributes by using this command.

Note: When modifying a WebSphere MQ server, it is important to understand the significance of the **name** and **serverName** attributes. For example, if a WebSphere Application Server administrator creates a WebSphere MQ server called “My European Area Server” that represents a WebSphere MQ queue manager with the **serverName** QM1, it would then be possible for the administrator to create a second WebSphere MQ server called “My UK Country Server”, which also represents the same queue manager.

Target object

A selected WebSphere MQ server.

Required parameters

-name
The name of the WebSphere MQ server. The name is specified when creating the WebSphere MQ server definition.

Conditional parameters

None

Optional parameters

-serverName
Name of the queue manager or queue-sharing group. This is the name by which the queue manager or queue-sharing group is identified. The value is allocated by WebSphere MQ administration to that WebSphere MQ resource. The administrator always uses the name that is allocated by WebSphere MQ administration.

-host
New value for the **host** attribute. This value is the name or the IP address of the host to which a connection is established when communicating with a queue manager or queue-sharing group that this WebSphere MQ server represents. The value is a string and must be one of the following:

- Symbolic host name
- IPv4 address
- IPv6 address

-bindingsMode
New value for the **bindingsMode** attribute. This value determines whether bindings transport mode connections are used when connecting to a queue manager or queue-sharing group. Bindings mode connection is possible if the application server and the queue manager are on the same node, but connection is only allowed to a single queue manager, even if multiple queue managers exist on the same node. This parameter has two possible values:

TRUE Bindings mode is used if available. If you select this option and bindings mode is not available, the connections mechanism defaults to client transport mode.

FALSE
Client mode is always used.

The default is TRUE.

-port
New value for the WebSphere MQ **port** attribute. This value is the TCP/IP port number on which the queue manager or queue-sharing group that this WebSphere MQ server represents listens. The default value is 1414.

-channel
New value for the WebSphere MQ **channel** attribute. This value is the WebSphere MQ client channel name to use when connecting to the queue manager or queue sharing group that this WebSphere MQ server represents. This name is allocated by WebSphere MQ administration to the WebSphere MQ object, and must always be used by the WebSphere Application Server administrator. The default value is SYSTEM.DEF.SVRCONN.

-description
New value for the **description** attribute. This value is a short description of the WebSphere MQ server, and is used for administrative purposes only.

-securityAuthAlias
New value for the **securityAuthorizationAlias** attribute. This value is the authentication alias to use when connecting to a queue manager or queue-sharing group. This parameter is not the same as the discovery authentication alias.

-transportChain
New value for the **transportChain** attribute. This value is the outbound transport chain to use when establishing a connection with WebSphere MQ. The default value is OutboundBasicWMQClient.

-trustUserIds
New value for the **trustUserIds** attribute. Determines whether user IDs received in messages from WebSphere MQ are propagated into the message or not (that is, whether user IDs received as part of message data are used in the service integration bus). The application user ID is always set from the **jsAppUserId** RFH2 value. If this is not present (either because the key/value pair is not present in the RFH2 header, or because the message does not have a RFH2 header), this field is not set. If you set this value to FALSE, the user ID is overwritten with the WebSphere MQ server name. This parameter has two possible values:

TRUE User IDs are propagated into messages.

FALSE
User IDs are not propagated into messages.

The default is TRUE.

-allowDiscovery
New value for the **allowDiscovery** attribute. This value determines whether automated discovery of WebSphere MQ resources is performed. This parameter has two possible values:

TRUE Automatic resource discovery is enabled.

FALSE
Automatic resource discovery is disabled.

The default is TRUE.

-discoveryAuthAlias
New value for the **discoveryAuthAlias** attribute. This value is the authentication alias to use when establishing a resource discovery connection to a queue manager or queue-sharing group, and is not the same as the security authentication alias parameter.

-replyToQueue
New value for the **replyToQueue** attribute. This value is the reply-to queue to use for resource discovery, is the name allocated by WebSphere MQ administration to the WebSphere MQ object, and must be the name of a model queue for a temporary dynamic queue. WebSphere Application Server administration must always use the name that is agreed with WebSphere MQ administration. The default is SYSTEM.DEFAULT.MODEL.QUEUE.

Example

- Using Jython:

```
wsadmin>AdminTask.modifySIBWMQServer(["-name", "Finance dept QM",
 "-allowDiscovery", "false"] )
```

- Using Jacl:

```
wsadmin>$AdminTask modifySIBWMQServer {-name "Finance dept QM"
 -allowDiscovery false}
```

listSIBWMQServers command

Use the listSIBWMQServers command to list the WebSphere MQ servers known to a cell.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command lists the WebSphere MQ servers known to the cell.

Target object

The current cell.

Required parameters

None

Conditional parameters

None

Optional parameters

None

Example

- Using Jython:

```
wsadmin>AdminTask.listSIBWMQServers()
```

- Using Jacl:

```
wsadmin>$AdminTask listSIBWMQServers
```

showSIBWMQServer command

Use the showSIBWMQServer command to show details of a WebSphere MQ server.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBAdminCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command shows the details of a WebSphere MQ server.

Target object

A selected WebSphere MQ server.

Required parameters

-name
The name of the WebSphere MQ server. This value is the name that you specified in the **-name** parameter when creating the WebSphere MQ server.

Conditional parameters

None

Optional parameters

None

Example

- Using Jython:

```
AdminTask.showSIBWMQServer(["-name", "Finance dept QM"])
```

- Using Jacl:

```
$AdminTask showSIBWMQServer {-name "Finance dept QM"}
```

deleteSIBWMQServer command

Use the deleteSIBWMQServer command to delete a specified WebSphere MQ server and remove the associated WebSphere MQ server bus members, assigned queue points and mediation points.

You can delete a WebSphere MQ server by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting a WebSphere MQ server definition” on page 576.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command deletes a specified WebSphere MQ server and removes any associated WebSphere MQ server bus members, assigned queue points and mediation points. This operation does not remove the corresponding WebSphere MQ queue manager or queue-sharing group, and it does not remove messages from any queues that are assigned to corresponding WebSphere MQ server bus members. The bus members for a given WebSphere MQ server are those created when the server is added to a bus.

Target object

A selected WebSphere MQ server.

Required parameters

-name
The name of the WebSphere MQ server to delete. This value is specified for the **-name** parameter (not the **-serverName** parameter), when the WebSphere MQ server is created, and is the name of the WebSphere Application Server administrative object, not the name of the WebSphere MQ object. If you issue the delete command, it is the WebSphere Application Server administrative object that is deleted, not the WebSphere MQ resource.

Conditional parameters

None

Optional parameters

None

Example

- Using Jython:

```
AdminTask.deleteSIBWMQServer(["-name", "Finance dept QM"])
```

- Using Jacl:

```
$AdminTask deleteSIBWMQServer {-name "Finance dept QM"}
```

modifySIBWMQServerBusMember command

Use the modifySIBWMQServerBusMember command to modify the attributes of a WebSphere MQ server bus member.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBAdminCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A WebSphere MQ server bus member is used for assigning queue points and mediation points to WebSphere MQ queues. This command modifies the attributes of a WebSphere MQ server bus member.

Target object

A selected WebSphere MQ server bus member.

Required parameters

-name
The administrative name of the WebSphere MQ server bus member. This value is the name specified in the **-name** parameter when creating the WebSphere MQ server definition.

-bus
The name of the service integration bus of which the WebSphere MQ server is a member.

Conditional parameters

None

Optional parameters

-virtualQueueManagerName
When sending messages to WebSphere MQ, the WebSphere MQ gateway queue manager sees the bus as a remote queue manager. The virtual queue manager name is the name that is passed to WebSphere MQ as the name of this remote queue manager.

The default value is the name of the Service Integration bus. If this bus name is not a valid name for a WebSphere MQ queue manager, or if another WebSphere MQ queue manager already has the same name, replace the default value with a unique, valid name for a WebSphere MQ queue manager.

- It must contain between 1 and 48 characters.
- It must conform to the WebSphere MQ queue naming rules (see the *Rules for naming WebSphere MQ objects* topic in the WebSphere MQ information center).

-host
The new value for the overridden **host** attribute. This value is the name or the IP address of the host to which a connection is established for communicating with a queue manager or queue-sharing group that this WebSphere MQ server represents. The value is a string and must be one of the following:

- Symbolic host name
- IPv4 address
- IPv6 address

-port
The new value for the overridden **port** attribute. This value is the TCP/IP port number on which the queue manager or queue-sharing group that this WebSphere MQ server represents listens. The default value is 1414.

-channel
The new value for the overridden channel attribute. This value is the WebSphere MQ client channel name to use when connecting to the queue manager or queue-sharing group that this WebSphere MQ server represents. The default value is SYSTEM.DEF.SVRCONN.

-securityAuthAlias
The new value for the overridden **securityAuthAlias** attribute. This value is the authentication alias to use when connecting to a queue manager or queue-sharing group. This parameter is not the same as the discovery authentication alias.

-transportChain
The new value for the overridden **transportChain** attribute. This value is the outbound transport chain to use when establishing a connection with WebSphere MQ. The default value is OutboundBasicWMQClient.

-trustUserIds TRUE | FALSE
The new value for the overridden **trustUserIds** attribute. This value determines whether user identifiers that are received in messages from WebSphere MQ are propagated into the message (that is, whether user identifiers that are received as part of message data are used in the service integration bus). The application user identifier is always set from the **jsAppUserId** RFH2 value. If this is not present (either because the key/value pair is not present in the RFH2 header, or because the message does not have a RFH2 header), this field is not set. If you set this value to FALSE, the user identifier is overwritten with the WebSphere MQ server name. This parameter has two possible values:

TRUE User identifiers are propagated into messages.

FALSE
User identifiers are not propagated into messages.

The default is TRUE.

Example

- Using Jython:

```
wsadmin>AdminTask.modifySIBWMQServerBusMember(["-name", "Finance dept QM-Bus1",
"-bus", "Bus1", "-trustUserIds", "false"])
```

- Using Jacl:

```
wsadmin>$AdminTask modifySIBWMQServerBusMember {-name "Finance dept QM-Bus1"
 -bus Bus1 -trustUserIds false}
```

listSIBWMQServerBusMembers command

Use the listSIBWMQServerBusMembers command to list the WebSphere MQ server bus members known to the bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command lists the WebSphere MQ server bus members known to the bus.

Target object

The current cell.

Required parameters

-bus
The name of the service integration bus of which the WebSphere MQ server is a member.

Conditional parameters

None

Optional parameters

None

Example

- Using Jython:
  ```
  AdminTask.listSIBWMQServerBusMembers(["-bus", "Bus1"])
  ```
- Using Jacl:
  ```
  $AdminTask listSIBWMQServerBusMembers {-bus Bus1}
  ```

showSIBWMQServerBusMember command

Use the showSIBWMQServerBusMember command to show a list of attributes for a WebSphere MQ server bus member.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

A WebSphere MQ server represents a WebSphere MQ queue manager or (for WebSphere MQ for z/OS) queue-sharing group. This command shows details of a WebSphere MQ server bus member as a set of key/value pairs. The command shows details for those attributes that have a value.

Target object

A selected WebSphere MQ server.

Required parameters

-name
The name of the WebSphere MQ server bus member. This value is the name specified in the -name parameter when creating the WebSphere MQ server definition.

-bus
The name of the bus for which the WebSphere MQ server is a member.

Conditional parameters

None

Optional parameters

None

Example

- Using Jython:

  ```
  wsadmin>AdminTask.showSIBWMQServerBusMember(["-name", "Finance Dept. QM-Bus1",
   "-bus", "Bus1"])
  ```

- Using Jacl:

  ```
  wsadmin>$AdminTask showSIBWMQServerBusMember {-name "Finance Dept. QM-Bus1"
   -bus Bus1}
  ```

SIBAdminBusSecurityCommands command group for the AdminTask object

You can use these administrative commands to manage service integration bus security.

These commands provide an alternative to using the administrative console or using the more complex syntax of wsadmin and JACL.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

populateUniqueNames command

Use the populateUniqueNames command to add missing unique names to the authorization policy for a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The populateUniqueNames command queries the user repository for missing unique names, and adds the missing unique names to the authorization policy for a selected service integration bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

-force TRUE | FALSE
Whether to update all unique names, or only those that are missing in the authorization policy. This parameter has two possible values:

TRUE All unique names are updated.

FALSE
Missing unique names only are updated.

The default is FALSE.

Example

The following example queries the user repository for `Bus1` for unique names, and updates all the unique names in the authorization policy.

```
AdminTask.populateUniqueNames('[-bus Bus1 -force TRUE]')
```

listGroupsInBusConnectorRole command

Use the listGroupsInBusConnectorRole command to list all the groups in the connector role for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listGroupsInBusConnectorRole command to list the groups in the bus connector role for a local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default is FALSE.

Example

The following example lists all the groups in the bus connector role for a bus called `Bus1`.

```
AdminTask.listGroupsInBusConnectorRole ('[-bus Bus1]')
```

addGroupToBusConnectorRole command

Use the addGroupToBusConnectorRole command to add a group to the connector role for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addGroupToBusConnectorRole command to add a group to the bus connector role for a local bus. This command grants the group permission to connect to local destinations.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-group *groupName*
The name of a group you want to add to the bus connector role for the local bus. You can type a specific group name, or use one of the following specialized group names:

Server
This group contains application servers.

AllAuthenticated
This group contains authenticated users only.

Everyone
This group contains all users. Each user is anonymous.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the group in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the group. You can specify values for both **-uniqueName** and **-group**, but you must ensure that they identify the same group. The command does not check that the values match.

Examples

The following example adds a group with the group name `Group1`, and the unique name `Group1` to the bus connector role for a bus called `Bus1`.

```
AdminTask.addGroupToBusConnectorRole ('[-bus Bus1  -group Group1 -uniqueName SalesGroup]')
```

The following example adds the `Server` group to the bus connector role for a bus called `Bus1`.

```
AdminTask.addGroupToBusConnectorRole ('[-bus Bus1  -group Server]')
```

The following example adds the `AllAuthenticated` group to the bus connector role for a bus called `Bus1`.

```
AdminTask.addGroupToBusConnectorRole ('[-bus Bus1  -group AllAuthenticated]')
```

The following example adds the `Everyone` group to the bus connector role for a bus called `Bus1`.

```
AdminTask.addGroupToBusConnectorRole ('[-bus Bus1  -group Everyone]')
```

removeGroupFromBusConnectorRole command

Use the removeGroupFromBusConnectorRole command to remove a group from the connector role for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeGroupFromBusConnectorRole command to remove a group from the bus connector role for a local bus. Removing a group from this role prevents the group from accessing local destinations, or sending messages to destinations on foreign buses.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-group *groupName* | *uniqueName*
The name of a group you want to remove from the bus connector role for the local bus. You can type one of the following names:

- A security group name, or one of the following specialized group names:

  Server
  This group contains application servers.

  AllAuthenticated
  This group contains authenticated users only.

  Everyone
  This group contains all users. Each user is anonymous.

- A unique group name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a group called `Group1` from the bus connector role for a bus called `Bus1`.

```
AdminTask.removeGroupFromBusConnectorRole ('[-bus Bus1  -group Group1]')
```

The following example removes the `Server` group from the bus connector role for a bus called `Bus1`.

```
AdminTask.removeGroupFromBusConnectorRole ('[-bus Bus1  -group Server]')
```

listUsersInBusConnectorRole command

Use the listUsersInBusConnectorRole command to list users in the connector role for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listUsersInBusConnectorRole command to list all the users in the bus connector role for a selected local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default is FALSE.

Example

The following example lists users in the bus connector role for a bus called `Bus1`.

```
AdminTask.listUsersInBusConnectorRole ('[-bus Bus1]')
```

addUserToBusConnectorRole command

Use the addUserToBusConnectorRole command to add a user to the connector role for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addUserToBusConnectorRole command to add a user to the bus connector role for a local bus. This command grants the user permission to connect to local destinations.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-user *userName*
The name of a user you want to add to the bus connector role for the local bus.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the user in the user registry. If an LDAP user registry is in use,

the unique name is the distinguished name (DN) for the user. You can specify values for both **-uniqueName** and **-user**, but you must ensure that they identify the same user. The command does not check that the values match.

Examples

The following example adds a user called `User1` to the bus connector role for a bus called `Bus1`.

```
AdminTask.adduserToBusConnectorRole ('[-bus Bus1  -user User1]')
```

removeUserFromBusConnectorRole command

Use the removeUserFromBusConnectorRole command to remove a user from the connector role for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeUserFromBusConnectorRole command to remove a user from the bus connector role for a local bus. Removing a user from this role prevents the user from accessing local destinations, or sending messages to destinations on foreign buses.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-user *userName* **or** *uniqueName*
The name of a user you want to remove from the bus connector role for the local bus. You can type one of the following names:

- A security user name.
- A unique user name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a user called `User1` from the bus connector role for a bus called `Bus1`.

```
AdminTask.removeUserFromBusConnectorRole ('[-bus Bus1  -user User1]')
```

The following example removes the `Server` user from the bus connector role for a bus called `Bus1`.

```
AdminTask.removeUserFromBusConnectorRole ('[-bus Bus1  -user Server]')
```

listGroupsInDefaultRole command

Use the listGroupsInDefaultRole command to list the groups in the default roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listGroupsInDefaultRole command to list the groups in the default roles for a selected local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
The role type for which you want to list groups.

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

Creator
This role type is authorized to create messages on destinations on the local bus.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default is FALSE.

Examples

The following example lists the groups in the `Sender` role for a bus called `Bus1`.

```
AdminTask.listGroupsInDefaultRole ('[-bus Bus1 -role Sender]')
```

addGroupToDefaultRole command

Use the addGroupToDefaultRole command to add a group to the default roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addGroupToDefaultRole command to grant a group default access to all local bus destinations for the specified roles. Adding a group to the default role does not grant access to local destinations where the inheritance of default access is disallowed. To grant access to a local destination where inheritance is disallowed, you must add the group to a destination role. For more information, see "addGroupToDestinationRole command" on page 2398.

You can use this command to define the access control policy for a messaging resource that does not yet exist. This approach ensures that the messaging resource is secure from the moment it is created.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
The role type to which you want to assign the group. You can assign a group to the following role types:

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

Creator
This role type is authorized to create messages on destinations on the local bus.

-group *groupName*
The name of a group you want to add to default roles for the local bus. You can type a specific group name, or use one of the following specialized group names:

Server
This group contains application servers.

AllAuthenticated
This group contains authenticated users only.

Everyone
This group contains all users. Each user is anonymous.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the group in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the group. You can specify values for both **-uniqueName** and **-group**, but you must ensure that they identify the same group. The command does not check that the values match.

Examples

The following example adds a group with the group name `Group1`, and the unique name `SalesGroup`, to the sender role type for a bus called `Bus1`.

```
AdminTask.addGroupToDefaultRole ('[-bus Bus1 -role Sender -group Group1 uniqueName SalesGroup]')
```

The following example adds the `AllAuthenticated` group to the browser role for a bus called `Bus1`.

```
AdminTask.addGroupToDefaultRole ('[-bus Bus1 -role Browser -group AllAuthenticated]')
```

removeGroupfromDefaultRole command

Use the removeGroupFromDefaultRole command to remove a group from the default roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeGroupFromDefaultRole command to remove a group from the default roles for a local bus. Removing a group from the default roles prevents access to the local bus in the default roles.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
The role type from which you want to remove the group.

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

Creator
This role type is authorized to create messages on destinations on the local bus.

-group *groupName* **or** *uniqueName*
The name of a group you want to remove from the default roles for the local bus. You can type one of the following names:

- A security group name, or one of the following specialized group names:

  Server
  This group contains application servers.

  AllAuthenticated
  This group contains authenticated users only.

  Everyone
  This group contains all users. Each user is anonymous.
- A unique group name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a group called `Group1` from the default `Sender` role type for a bus called `Bus1`.

```
AdminTask.removeGroupFromDefaultRole ('[-bus Bus1 -role Sender -group Group1]')
```

The following example removes the `AllAuthenticated` group from the default browser role for a bus called `Bus1`.

```
AdminTask.removeGroupFromDefaultRole ('[-bus Bus1 -role Browser -group AllAuthenticated]')
```

listUsersInDefaultRole command

Use the listUsersInDefaultRole command to list users in the default roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listUsersInDefaultRole command to list all the users in the default roles for a selected local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
The role type for which you want to list users. You can specify the following role types:

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

Creator
This role type is authorized to create messages on destinations on the local bus.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default value is FALSE.

Example

The following example lists users in the `Sender` role type for a bus called `Bus1`.

```
AdminTask.listUsersInDefaultRole ('[-bus Bus1 -role Sender]')
```

addUserToDefaultRole command

Use the addUserToDefaultRole command to add a user to the default roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addUserToDefaultRole command to grant a user default access to all local bus destinations for the specified roles. Adding a user to the default role does not grant access to local destinations where the inheritance of default access is disallowed. To grant access to a local destination where inheritance is disallowed, you must add the user to a destination role. For more information, see “addUserToDestinationRole command” on page 2403.

You can use this command to define the access control policy for a messaging resource that does not yet exist. This approach ensures that the messaging resource is secure from the moment it is created.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
The role type to which you want to assign the user. You can assign a user to the following role types:

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

Creator
This role type is authorized to create messages on destinations on the local bus.

-user *userName*
The name of a user you want to add to the bus connector role for the local bus.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the user in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the user. You can specify values for both **-uniqueName** and **-user**, but you must ensure that they identify the same user. The command does not check that the values match.

Examples

The following example adds a user called `User1` to the sender role type for a bus called `Bus1`.

```
AdminTask.addUserToDefaultRole ('[-bus Bus1 -role Sender -user User1]')
```

removeUserfromDefaultRole command

Use the removeGroupFromDefaultRole command from remove a user from the default roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeUserFromDefaultRole command to remove user from default roles for a local bus. Removing a user from the default roles prevents access to the local bus in the default roles.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
The role type you want to remove the user from. You can remove a user from the following role types:

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

Creator
This role type is authorized to create messages on destinations on the local bus.

-user *userName* **or** *uniqueName*
The name of a user you want to remove from the default roles for the local bus. You can type one of the following names:

- A security user name.
- A unique user name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a user called `User1` from the `Sender` role type for a bus called `Bus1`.

```
AdminTask removeUserFromDefaultRole ('[-bus Bus1 -role Sender -User User1]')
```

listGroupsInTopicRole command

Use the listGroupsInTopicRole command to list the groups in topic roles for a topic space on a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listGroupsInTopicRole command to list the groups in topic roles for a selected topic in the topic hierarchy for a selected local bus.

You can use this command to define the access control policy for a topic that does not yet exist.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-topic *topicName*
The name of the topic.

-role *roleType*
You can specify the Sender or Receiver roles for a topic.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default value is FALSE.

Examples

The following example lists the groups in the `Sender` role for a topic called `football`, in the topic space called `Sport`, on a local bus called `Bus1`.

```
listGroupsInTopicRole { -bus Bus1 -topicSpace Sport -topic football -role Sender}
```

addGroupToTopicRole command

Use the addGroupToTopicRole command to add a group to a topic role within a specified topic space.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addGroupToTopicRole command to add a group to the sender or receiver roles for a topic anywhere in the topic hierarchy for a local bus. The roles that you specify for the topic are additional to any roles that the topic inherits from its parent in the topic hierarchy.

You can use this command to define the access control policy for a topic that does not yet exist. By defining the access control policy first, you ensure that the topic is secure from the moment it is created.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-topic *topicName*
The name of the topic.

-role *roleName*
You can specify the Sender or Receiver roles for a topic.

-group *groupName*
The name of the group that you want to add to the sender or receiver roles for a topic. You can specify a group name, or one of the following specialized group names:

Server
This group contains application servers.

AllAuthenticated
This group contains authenticated users only.

Everyone
This group contains all users. Each user is anonymous.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the group in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the group. You can specify values for both **-uniqueName** and **-group**, but you must ensure that they identify the same group. The command does not check that the values match.

Examples

The following example adds a group with the group name `Group1`, and the unique name `SalesGroup`, to the `Sender` role for a topic called `football`, in the topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.addGroupToTopicRole ('[-bus Bus1 -topicSpace Sport
 -topic football -role Sender -group Group1 -uniqueName SalesGroup]')
```

removeGroupFromTopicRole command

Use the removeGroupFromTopicRole command to remove a group from a topic role within a specified topic space.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeGroupFromTopicRole command to remove a group from the sender or receiver roles for a topic anywhere in the topic hierarchy for a local bus. Removing a group from topic roles prevents the group from accessing the topic space in the topic roles.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-topic *topicName*
The name of the topic.

-role *roleName*
You can specify the Sender or Receiver roles for a topic.

-group *groupName* **or** *uniqueName*
The name of a group you want to remove from the topic roles for the local bus. You can type one of the following names:

- A security group name, or one of the following specialized group names:

  Server
  This group contains application servers.

  AllAuthenticated
  This group contains authenticated users only.

  Everyone
  This group contains all users. Each user is anonymous.
- A unique group name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a group called `Group1`, from the `Sender` role for a topic called `football`, in the topic space called `Sport`, on a local bus called `Bus1`.

```
removeGroupFromTopicRole { -bus Bus1 -topicSpace Sport -topic football -role Sender -group Group1}
```

listUsersInTopicRole command

Use the listUsersInTopicRole command to list users in the topic roles within a specified topic space.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listUsersInTopicRole command to list users in the topic roles for a selected topic in the topic hierarchy for a selected local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-topic *topicName*
The name of the topic.

-role *roleType*
You can specify the Sender or Receiver roles for a topic.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default is FALSE.

Example

The following example lists the users in the `Sender` role for a topic called `football`, in the topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.listUsersInTopicRole ('[-bus Bus1 -topicSpace Sport -topic football -role Sender]')
```

addUserToTopicRole command

Use the addUserToTopicRole command to add a user to a topic role within a specified topic space.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addUserToTopicRole command to add a user to the sender or receiver roles for a topic anywhere in the topic hierarchy for a local bus. The roles that you specify for the topic are additional to any roles that the topic inherits from its parent in the topic hierarchy.

You can use this command to define the access control policy for a topic that does not yet exist. By defining the access control policy first, you ensure that the topic is secure from the moment it is created.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-topic *topicName*
The name of the topic.

-role *roleName*
You can specify the Sender or Receiver roles for a topic.

-user *userName*
The name of the user that you want to add to the Sender or Receiver roles for a topic.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the user in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the user. You can specify values for both **-uniqueName** and **-user**, but you must ensure that they identify the same user. The command does not check that the values match.

Examples

The following example adds a user called `User1`, to the `Sender` role for a topic called `football`, in the topic space called `Sport`, on a local bus called `Bus1`.

```
Admintask.addUserToTopicRole ('[-bus Bus1 -topicSpace Sport -topic football -role Sender -user User1]')
```

removeUserFromTopicRole command

Use the removeUserFromTopicRole command to remove a user from a topic role within a specified topic space.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeUserFromTopicRole command to remove a user from the sender or receiver roles for a topic anywhere in the topic hierarchy for a local bus. Removing a user from topic roles prevents the user from accessing the topic space in the topic roles.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-topic *topicName*
The name of the topic.

-role *roleType*
You can specify the Sender or Receiver roles for a topic.

-user *userName* **or** *uniqueName*
The name of a user you want to remove from the topic roles for the local bus. You can type one of the following names:

- A security user name.
- A unique user name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a user called `User1`, from the `Sender` role for a topic called `footbal`l, in the topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.removeUserFromTopicRole ('[-bus Bus1 -topicSpace Sport
 -topic football -role Sender -user User1]')
```

listGroupsInTopicSpaceRootRole command

Use the listGroupInTopicSpaceRootRole command to list all the groups in topic space roles on the topic space root for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listGroupsInTopicRootSpaceRole command to list groups in topic space root roles for a selected topic space on a selected local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-role *roleType*
You can specify the Sender or Receiver roles for a topic.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default is FALSE.

Examples

The following example lists groups in the `Sender` role for the topic space root, for a topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.listGroupsInTopicSpaceRootRole ('[-bus Bus1 -topicSpace Sport -role Sender]')
```

addGroupToTopicSpaceRootRole command

Use the addGroupToTopicSpaceRootRole command to add a group to topic space roles on the topic space root.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addGroupToTopicRootSpaceRole command to grant a group permission to access the topic space root in the sender and receiver roles. This is in addition to any access permissions granted to the topics in the topic space.

You can use this command to define the access control policy for a topic that does not yet exist. By defining the access control policy first, you ensure that the topic is secure from the moment it is created.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-role *roleName*
You can specify the Sender or Receiver roles for a topic.

-group *groupName*
The name of the group that you want to add to the Sender or Receiver roles for the topic space root. You can specify a group name, or one of the following specialized group names:

Server
This group contains application servers.

AllAuthenticated
This group contains authenticated users only.

Everyone
This group contains all users. Each user is anonymous.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the group in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the group. You can specify values for both **-uniqueName** and **-group**, but you must ensure that they identify the same group. The command does not check that the values match.

Examples

The following example adds a group called `Group1`, and the unique name `SalesGroup` to the `Sender` role for the topic space root, for a topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.addGroupToTopicSpaceRootRole ('[-bus Bus1 -topicSpace Sport
 -role Sender -group Group1 uniqueName SalesGroup]')
```

removeGroupFromTopicSpaceRootRole command

Use the removeGroupFromTopicSpaceRootRole command to remove a group from access roles on the topic space root.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeGroupFromTopicSpaceRootRole command to remove a group from the Sender and Receiver roles for the topic space root.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-role *roleName*
You can specify the Sender or Receiver roles for a topic.

-group *groupName* **or** *uniqueName*
The name of a group you want to remove from the topic space root roles for the local bus. You can type one of the following names:

- A security group name, or one of the following specialized group names:

  Server
  This group contains application servers.

  AllAuthenticated
  This group contains authenticated users only.

  Everyone
  This group contains all users. Each user is anonymous.

- A unique group name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a group called `Group1` from the `Sender` role for the topic space root, for a topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.removeGroupFromTopicSpaceRootRole ('[-bus Bus1 -topicSpace Sport -role Sender -group Group1]')
```

listUsersInTopicSpaceRootRole command

Use the listUsersInTopicSpaceRootRole command to list the users in the sender and receiver roles on the topic space root.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listUsersInTopicSpaceRootRole command to list users in the topic space root roles for a selected local bus.

Target object

None.

Required parameters

-bus *busName*

The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-role *roleType*
You can specify the Sender or Receiver roles for the topic space root.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default value is FALSE.

Example

The following example lists users in the `Sender` role for the topic space root, for a topic space called `Sport`, on a local bus called `Bus1`.

```
Admin.TasklistUsersInTopicSpaceRootRole { -bus Bus1 -topicSpace Sport -role Sender}
```

addUserToTopicSpaceRootRole command

Use the addUserToTopicSpaceRootRole command to add a user to the sender and receiver roles on the topic space root.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addUserToTopicSpaceRootRole command to grant a user permission to access the topic space root in the sender and receiver roles. This is in addition to any access permissions granted to the topics in the topic space.

You can use this command to define the access control policy for a topic that does not yet exist. By defining the access control policy first, you ensure that the topic is secure from the moment it is created.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-role *roleName*
You can specify the Sender or Receiver roles for the topic space root.

-user *userName*
The name of the user that you want to add to the Sender or Receiver roles for the topic space root.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the user in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the user. You can specify values for both **-uniqueName** and **-user**, but you must ensure that they identify the same user. The command does not check that the values match.

Examples

The following example adds a user called `User1`, to the `Sender` role for the topic space root, for a topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.addUserToTopicSpaceRootRole  ('[-bus Bus1 -topicSpace Sport -role Sender -user User1]')
```

removeUserFromTopicSpaceRootRole command

Use the removeUserFromTopicSpaceRootRole command to remove a user from access roles on the topic space root.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeUserFromTopicSpaceRootRole command to remove a user from the Sender and Receiver roles for the topic space root.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-topicSpace *topicSpaceName*
The name of the topic space.

-role *roleType*
You can specify the Sender or Receiver roles for a topic.

-user *userName* **or** *uniqueName*
The name of a user you want to remove from the topic space root roles for the local bus.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a user called `User1` from the `Sender` role for the topic space root, for a topic space called `Sport`, on a local bus called `Bus1`.

```
AdminTask.removeUserFromTopicSpaceRootRole ('[-bus Bus1 -topicSpace Sport -role Sender -user User1]')
```

listGroupsInDestinationRole command

Use the listGroupsInDestinationRole command to list the groups in the destination roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listGroupsInDestinationRole command to list all the groups in destination roles for a selected local bus. The roles you can specify depend on the type of destination.

Target object

None.

Required parameters

-type *destinationType*

You can specify one of the following destination types:

- Queue
- Port
- TopicSpace
- ForeignDestination
- Alias

The allowed roles for a destination depend on the type of the destination as defined in “Administering destination roles” on page 1991.

If you are specifying a *destinationType* that is either foreignDestination or alias, the foreign bus name that you specify must be the name of the foreign bus hosting the destination.

If you specify a *destinationType* of queue or topic, the foreign bus name is ignored. The authorization is granted against the destination in the local bus.

-bus *busName*

The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*

You can specify one of the following role types, depending on the **-type** you have specified.

Sender

This role type is authorized to send messages to destinations on the local bus.

Receiver

This role type is authorized to receive messages from destinations on the local bus.

Browser

This role type is authorized to browse messages on destinations on the local bus.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default value is FALSE.

Examples

The following example lists groups in the `Sender` role for a queue type destination called `Queue1`, on a local bus called `Bus1`.

```
AdminTask.listGroupsInDestinationRole ('[-type queue -bus Bus1 -destination Queue1 -role Sender]')
```

The following example lists groups in the `Receiver` role for a queue type destination called `Queue2`, on a local bus called `Bus1`.

```
AdminTask.listGroupsInDestinationRole ('[-type queue -bus BusName -destination Queue2 -role Receiver]')
```

---

addGroupToDestinationRole command

Use the addGroupToDestinationRole command to add a group to the destination roles for a local or foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addGroupToDestinationRole command to grant a group access to local bus destinations for the specified roles. The roles you can specify depend on the type of destination.

Target object

None.

Required parameters

-type *destinationType*
You can specify one of the following destination types:

- Queue
- Port
- TopicSpace
- ForeignDestination
- Alias

The allowed roles for a destination depend on the type of the destination as defined in “Administering destination roles” on page 1991.

If you are specifying a *destinationType* that is either foreignDestination or alias, the foreign bus name that you specify must be the name of the foreign bus hosting the destination.

If you specify a *destinationType* of queue or topic, the foreign bus name is ignored. The authorization is granted against the destination in the local bus.

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
You can specify one of the following role types, depending on the **-type** you have specified.

Sender
This role type applies to alias, foreignDestination, port, queue, and topicSpace destination types.

Receiver
This role type applies to alias, port, queue, and topicSpace destination types.

Browser
This role type applies to alias, port, and queue destination types.

-group *groupName*
The name of the group that you want to add to the destination role type for the local bus. You can specify a group name, or one of the following specialized group names:

Server
This group contains application servers.

AllAuthenticated
This group contains authenticated users only.

Everyone
This group contains all users. Each user is anonymous.

Conditional parameters

None.

Optional parameters

-foreignBus *foreignBusName*
Specify the name of the foreign bus. If you are adding a group to a destination on a foreign destination or an alias, you must specify the name of the foreign bus that hosts the foreign destination or the alias.

-uniqueName *uniqueName*
This parameter is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions. Specify the name that uniquely defines the group in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the group. You can specify values for both **-uniqueName** and **-group**, but you must ensure that they identify the same group. The command does not check that the values match.

Examples

The following example adds a group with the group name `Group1`, and the unique name `SalesGroup` to the sender role on a queue type destination called `Queue1`, on a local bus called `Bus1`.

```
AdminTask.addGroupToDestinationRole ('[-type queue -bus Bus1
 -destination Queue1 -role Sender -group Group1 -uniqueName SalesGroup]')
```

The following example adds a group called `Group2` to the receiver role on a queue type destination called `Queue2`, on a local bus called `Bus1`.

```
AdminTask.addGroupToDestinationRole ('[-type queue -bus Bus1
 -destination Queue2 -role Receiver -group Group2]')
```

removeGroupFromDestinationRole command

Use the removeGroupFromDestinationRole command to remove a group from the destination roles for a local or foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:

  `print AdminTask.help('`*command_name*`')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeGroupFromDestinationRole command remove a group from destination roles for a selected local bus. By removing a group from destination roles, you prevent the group from accessing the local bus.

Target object

None.

Required parameters

-type *destinationType*

You can specify one of the following destination types:

- Queue
- Port
- TopicSpace
- ForeignDestination
- Alias

The allowed roles for a destination depend on the type of the destination as defined in “Administering destination roles” on page 1991.

If you are specifying a *destinationType* that is either foreignDestination or alias, the foreign bus name that you specify must be the name of the foreign bus hosting the destination.

If you specify a *destinationType* of queue or topic, the foreign bus name is ignored. The authorization is granted against the destination in the local bus.

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
You can specify one of the following role types, depending on the *destinationType* you have specified.

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

-group *groupName* **or** *uniqueName*
The name of a group you want to remove from the destination roles for the local bus. You can type one of the following names:

- A security group name, or one of the following specialized group names:

  Server
  This group contains application servers.

  AllAuthenticated
  This group contains authenticated users only.

  Everyone
  This group contains all users. Each user is anonymous.
- A unique group name.

Conditional parameters

None.

Optional parameters

-foreignBus *foreignBusName*
Specify the name of the foreign bus. If you are removing a group from a destination role on a foreign destination or an alias, you must specify the name of the foreign bus that hosts the foreign destination or the alias.

Examples

The following example removes a group called `Group1` from the `Sender` role for a queue type destination called `Queue1`, on a local bus called `Bus1`.

```
removeGroupFromDestinationRole { -type queue -bus Bus1
 -destination Queue1 -role Sender -group Group1}
```

The following example removes a group called `Group2` from the `Receiver` role for a queue type destination called `Queue2`, on a local bus called `Bus1`.

```
removeGroupFromDestinationRole { -type queue -bus Bus1
 -destination Queue2 -role Receiver -group Group2}
```

listUsersInDestinationRole command

Use the listUsersInDestinationRole command to list users in the destination roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listUsersInDestinationRole command list the users in destination roles for a selected local bus. The roles you can specify depend on the type of destination.

Target object

None.

Required parameters

-type *destinationType*

You can specify one of the following destination types:

- Queue
- Port
- TopicSpace
- ForeignDestination
- Alias

The allowed roles for a destination depend on the type of the destination as defined in “Administering destination roles” on page 1991.

If you are specifying a *destinationType* that is either foreignDestination or alias, the foreign bus name that you specify must be the name of the foreign bus hosting the destination.

If you specify a *destinationType* of queue or topic, the foreign bus name is ignored. The authorization is granted against the destination in the local bus.

-bus *busName*

The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*

You can specify one of the following role types, depending on the **-type** you have specified.

Sender
This role type applies to alias, foreignDestination, port, queue, and topicSpace destination types.

Receiver
This role type applies to alias, port, queue, and topicSpace destination types.

Browser
This role type applies to alias, port, and queue destination types.

Conditional parameters

None.

Optional parameters

-foreignBus *foreignBusName*
Specify the name of the foreign bus if you have specified a ForeignDestination or Alias destination type.

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default is FALSE.

Example

The following example lists users in the `Sender` role on a queue type destination called `Queue1`, on a local bus called `Bus1`.

```
AdminTask.listUsersInDestinationRole ('[-type queue -bus Bus1 -destination Queue1 -role Sender]')
```

addUserToDestinationRole command

Use the addUserToDestinationRole command to add a user to the destination roles for a local or foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addUserToDestinationRole command grant a user access to local bus destinations for the specified roles. The roles you can specify depend on the type of destination.

Target object

None.

Required parameters

-type *destinationType*
You can specify one of the following destination types:

- Queue
- Port
- TopicSpace
- ForeignDestination
- Alias

The allowed roles for a destination depend on the type of the destination as defined in “Administering destination roles” on page 1991.

If you are specifying a *destinationType* that is either foreignDestination or alias, the foreign bus name that you specify must be the name of the foreign bus hosting the destination.

If you specify a *destinationType* of queue or topic, the foreign bus name is ignored. The authorization is granted against the destination in the local bus.

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
You can specify one of the following role types, depending on the **-type** you have specified.

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

-user *userName*
The name of the user that you want to add to the destination role type for the local bus.

Conditional parameters

None.

Optional parameters

-foreignBus *foreignBusName*
Specify the name of the foreign bus. If you are adding a user to a destination on a foreign destination or an alias, you must specify the name of the foreign bus that hosts the foreign destination or the alias.

-uniqueName *uniqueName*
This parameter is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions. Specify the name that uniquely defines the user in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name

(DN) for the user. You can specify values for both **-uniqueName** and **-user**, but you must ensure that they identify the same user. The command does not check that the values match.

Examples

The following example adds a user called `User1` to the sender role on a queue type destination called `Queue1`, on a local bus called `Bus1`.

```
Admintask.addUserToDestinationRole ('[-type queue -bus Bus1
 -destination Queue1 -role Sender -user User1]')
```

The following example adds a user called `User2` to the receiver role on a queue type destination called `Queue2`, on a local bus called `Bus1`.

```
Admintask.addUserToDestinationRole ('[-type queue -bus Bus1
 -destination Queue2 -role Receiver -user User2]')
```

removeUserFromDestinationRole command

Use the removeUserFromDestinationRole command to remove a user from the destination roles for a local or foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeUserFromDestinationRole command to remove a user from destination roles for a selected local bus. By removing a user from destination roles, you prevent the user from accessing the local bus.

Target object

None.

Required parameters

-type *destinationType*
You can specify one of the following destination types:

- Queue
- Port
- TopicSpace
- ForeignDestination

- Alias

The allowed roles for a destination depend on the type of the destination as defined in “Administering destination roles” on page 1991.

If you are specifying a *destinationType* that is either foreignDestination or alias, the foreign bus name that you specify must be the name of the foreign bus hosting the destination.

If you specify a *destinationType* of queue or topic, the foreign bus name is ignored. The authorization is granted against the destination in the local bus.

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-role *roleType*
You can specify one of the following role types, depending on the **-type** you have specified.

Sender
This role type is authorized to send messages to destinations on the local bus.

Receiver
This role type is authorized to receive messages from destinations on the local bus.

Browser
This role type is authorized to browse messages on destinations on the local bus.

-user *userName* **or** *uniqueName*
The name of a user you want to remove from the destination roles for the local bus. You can type one of the following names:

- A security user name.
- A unique user name.

Conditional parameters

None.

Optional parameters

-foreignBus *foreignBusName*
Specify the name of the foreign bus. If you are removing a user from a destination role on a foreign destination or an alias, you must specify the name of the foreign bus that hosts the foreign destination or the alias.

Examples

The following example removes a user called `User1` from the `Sender` role for a queue type destination called `Queue1`, on a local bus called `Bus1`.

```
AdminTask.removeUserFromDestinationRole ('[-type queue -bus Bus1
 -destination Queue1 -role Sender -user User1]')
```

The following example removes a user called `User2` from the `Receiver` role for a queue type destination called `Queue2`, on a local bus called `Bus1`.

```
AdminTask.removeUserFromDestinationRole ('[-type queue -bus Bus1
 -destination Queue2 -role Receiver -user User2]')
```

listGroupsInForeignBusRole command

Use the listGroupsInForeignBusRole command to list the groups in the sender role for a foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listGroupsInForeignBusRole command to list the groups that can send messages from a local bus to a foreign bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-foreignBus *foreignBusName*
The name of the foreign bus.

-role Sender
You can only specify the Sender role for a foreign bus.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default value is FALSE.

Examples

The following example lists the groups in the `Sender` role for a foreign bus called `ForeignBus1`. The local bus is called `Bus1`.

```
AdminTask.listGroupsInForeignBusRole ('[-bus Bus1 -foreignBus ForeignBus1 -role Sender]')
```

addGroupToForeignBusRole command

Use the addGroupToForeignBusRole command to grant a group permission to access a foreign bus from a local bus, in the sender role.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addGroupToForeignBusRole command to grant a group permission to send messages from a local bus to a foreign bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-foreignBus *foreignBusName*
The name of the foreign bus.

-role *Sender*
You can only specify the Sender role for a foreign bus.

-group *groupName*
The name of the group that you want to add to the sender role for the foreign bus. You can specify a group name, or one of the following specialized group names:

Server
This group contains application servers.

All Authenticated
This group contains authenticated users only.

Everyone
This group contains all users. Each user is anonymous.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the group in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the group. You can specify values for both **-uniqueName** and **-group**, but you must ensure that they identify the same group. The command does not check that the values match.

Examples

The following example adds a group with the group name `Group1`, and the unique name `SalesGroup` to the sender role for a foreign bus called `ForeignBus1`. The local bus is called `Bus1`.

```
AdminTask.addGroupToForeignBusRole ('[-bus Bus1 -ForeignBus ForeignBus1
 -role Sender -group Group1 -uniqueName SalesGroup]')
```

removeGroupFromForeignBusRole command

Use the removeGroupFromForeignBusRole command to remove a group from the sender role for a foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeGroupFromForeignBusRole command to remove a group from the sender role for a foreign bus. This prevents the group from sending messages from a local bus to a foreign bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-foreignBus *foreignBusName*
The name of the foreign bus.

-role Sender
You can only specify the Sender role for a foreign bus.

-group *groupName* **or** *uniqueName*
The name of a group you want to remove from the sender role for a foreign bus. You can type one of the following names:

- A security group name, or one of the following specialized group names:

  Server
  This group contains application servers.

  AllAuthenticated
  This group contains authenticated users only.

  Everyone
  This group contains all users. Each user is anonymous.
- A unique group name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a group called `Group1` from the `Sender` role for a foreign bus called `ForeignBus1`. The local bus is called `Bus1`.

```
AdminTask.removeGroupFromForeignBusRole ('[-bus Bus1 -ForeignBus ForeignBus1 -role Sender -group Group1]')
```

listUsersInForeignBusRole command

Use the listUsersInForeignBusRole command to list users that can send messages from a local bus to a foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the listUsersInForeignBusRole command to list all the users that can send messages from a selected local bus to a selected foreign bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-foreignBus *foreignBusName*
The name of the foreign bus.

-role Sender
You can specify the Sender role only for a foreign bus.

Conditional parameters

None.

Optional parameters

-showUniqueNames TRUE | FALSE
Whether to display unique names. This parameter has two possible values:

TRUE Unique names are displayed.

FALSE
Security names are displayed.

The default value is FALSE.

Example

The following example lists users in the `Sender` role for a foreign bus called `ForeignBus1`. The local bus is called `Bus1`.

```
AdminTask.listUsersInForeignBusRole ('[-bus Bus1 -foreignBus ForeignBus1 -role Sender],)
```

addUserToForeignBusRole command

Use the addUserToForeignBusRole command to grant a user permission to access a foreign bus from a local bus, in the sender role.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the addUserToForeignBusRole command grant a user permission to send messages from a local bus to a foreign bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-foreignBus *foreignBusName*
The name of the foreign bus.

-role Sender
You can specify the Sender role only for a foreign bus.

-user *userName*
The name of the user that you want to add to the Sender role for the foreign bus.

Conditional parameters

None.

Optional parameters

-uniqueName *uniqueName*
Specify the name that uniquely defines the user in the user registry. If an LDAP user registry is in use, the unique name is the distinguished name (DN) for the user. You can specify values for both **-uniqueName** and **-user**, but you must ensure that they identify the same user. The command does not check that the values match.

Examples

The following example adds a user called `User1` to the `Sender` role for a foreign bus called `ForeignBus1`. The local bus is called `Bus1`.

```
AdminTask.addUserToForeignBusRole ('[-bus Bus1 -foreignBus ForeignBus1 -role Sender -user User1]')
```

removeUserFromForeignBusRole command

Use the removeUserFromForeignBusRole command to remove a user from the sender role for a foreign bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeUserFromForeignBusRole command to remove a user from the sender role for a foreign bus. This prevents the user from sending messages from a local bus to a foreign bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-foreignBus *foreignBusName*
The name of the foreign bus.

-role Sender
You can only specify the Sender role for a foreign bus.

-user *userName* **or** *uniqueName*
The name of a user you want to remove from the sender role for a foreign bus. You can type one of the following names:

- A security user name
- A unique user name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a user called `User1` from the `Sender` role for a foreign bus called `ForeignBus1`. The local bus is called `Bus1`.

```
AdminTask.removeUserFromForeignBusRole ('[-bus Bus1 -ForeignBus ForeignBus1 -role Sender -user User1]')
```

removeGroupFromAllRoles command

Use the removeGroupFromAllRoles command to remove a group from all access roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeGroupFromAllRoles command to remove a group from all access roles for a local bus. This prevents the group from accessing the local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-group *groupName*
The name of the group that you want to remove from all roles for the bus. You can specify a group name, or one of the following specialized group names:

Server
This group contains application servers.

AllAuthenticated
This group contains authenticated users only.

Everyone
This group contains all users. Each user is anonymous.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a group called `Group1` from all roles for a local bus called `Bus1`.

```
removeGroupFromAllRoles { -bus Bus1 -group Group1}
```

removeUserFromAllRoles command

Use the removeUserFromAllRoles command to remove a user from all access roles for a local bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

Use the removeUserFromAllRoles command to remove a user from all access roles for a local bus. This prevents the user from accessing the local bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-user *userName* **or** *uniqueName*
The name of a user you want to remove from the bus connector role for the local bus. You can type one of the following names:

- A security group name.
- A unique group name.

Conditional parameters

None.

Optional parameters

None.

Examples

The following example removes a user called `User1` from all roles for a local bus called `Bus1`.

```
AdminTask.removeUserFromAllRoles ('[-bus Bus1 -user User1]')
```

Determining destination defaults inheritance by using the wsadmin tool

By default, the inheritance of default permissions by destinations is allowed for all local destinations. Use this command to determine whether the inheritance of default permissions is enabled or disabled for a local destination.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Command

To determine whether a specified destination inherits default destination user roles, use the following command:

- Using Jython:

  ```
  AdminTask.isInheritDefaultsForDestination("-type destinationType -bus busName
   -destination destinationName")
  ```

- Using Jacl:

  ```
  $AdminTask isInheritDefaultsForDestination {-type destinationType -bus busName
   -destination destinationName}
  ```

This command will return either `True` or `False`.

Defining destination defaults inheritance by using the wsadmin tool

By default, the inheritance of default permissions by destinations is allowed for all local destinations. Use this command to override inheritance for an individual destination, or to restore default inheritance in cases where you have previously overridden it.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Command

Note: When a user or group tries to access a destination for which you have overridden default inheritance, the default permissions are not checked. Only the permissions that you specify for the destination itself are checked.

To override or restore the inheritance of default permissions for a destination, use the following command:

- Using Jython:

  ```
  AdminTask.setInheritDefaultsForDestination("-type destinationType -bus busName
   -destination destinationName -inherit <true|false>")
  ```

- Using Jacl:

  ```
  $AdminTask setInheritDefaultsForDestination {-type destinationType -bus busName
   -destination destinationName -inherit <true|false>}
  ```

The *destinationType* must be a local destination type: *foreignDestination* is not allowed.

Set **-inherit** to false to override defaults inheritance, or to true to restore defaults inheritance.

Defining topic role inheritance by using the wsadmin tool

Use these commands to define the inheritance of topic roles by child topics within a topic hierarchy.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Commands

Defining Sender role inheritance

To set or disable Sender role inheritance for a topic within the specified topic space, use the following command:

- Using Jython:

  ```
  AdminTask.setInheritSenderForTopic("-bus busName
   -topicSpace topicSpaceName
   -topic topicName -inherit <true|false>")
  ```
- Using Jacl:

  ```
  $AdminTask setInheritSenderForTopic {-bus busName
   -topicSpace topicSpaceName
   -topic topicName -inherit <true|false>}
  ```

Defining Receiver role inheritance

To set or disable Receiver role inheritance for a topic within the specified topic space, use the following command:

- Using Jython:

  ```
  AdminTask.setInheritReceiverForTopic("-bus busName
   -topicSpace topicSpaceName -topic topicName
   -inherit <true|false>")
  ```
- Using Jacl:

  ```
  $AdminTask setInheritReceiverForTopic {-bus busName
   -topicSpace topicSpaceName
   -topic topicName -inherit <true|false>}
  ```

Determining topic role inheritance by using the wsadmin tool

Use these commands to list whether the child topics within a topic hierarchy currently have inheritance set or disabled.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Commands

Determining Inherit Receiver for Topic

To determine whether a role is inheritance Receiver for a topic within the specified topic space, use the following command:

- Using Jython:

```
AdminTask.isInheritReceiverForTopic("-bus busName -type typeName
 -destination destinationName")
```

- Using Jacl:

```
$AdminTask isInheritReceiverForTopic {-bus busName -type typeName
 -destination destinationName}
```

Determining Inherit Sender for Topic

To determine whether a role is inheritance Sender for a topic within the specified topic space, use the following command:

- Using Jython:

```
AdminTask.isInheritSenderForTopic("-bus busName -type typeName
 -destination destinationName")
```

- Using Jacl:

```
$AdminTask isInheritSenderForTopic {-bus busName -type typeName
 -destination destinationName}
```

Listing security roles for service integration by using the wsadmin tool

When you are administering messaging security, use these commands to list the security roles that are associated with service integration bus destinations, foreign buses, topics within a topic space, users and groups.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBAdminBusSecurityCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Commands

Listing all destinations with roles

To list all destinations that have roles defined for them, use the following command:

- Using Jython:

```
AdminTask.listAllDestinationsWithRoles("-bus busname
 -type destinationType")
```

- Using Jacl:

```
$AdminTask listAllDestinationsWithRoles {-bus busname
 -type destinationType}
```

Listing all foreign buses with roles

To list all foreign buses that have roles defined for them, use the following command:

- Using Jython:

```
AdminTask.listAllForeignBusesWithRoles("-bus busname")
```

- Using Jacl:

```
$AdminTask listAllForeignBusesWithRoles {-bus busname}
```

Listing all topics within a topic space with roles

To list all topics within a topic space that have roles defined for them, use the following command:

- Using Jython:

```
AdminTask.listAllTopicsWithRoles("-bus busname
 -topicSpace topicSpaceName")
```

- Using Jacl:

```
$AdminTask listAllTopicsWithRoles {-bus busname
 -topicSpace topicSpaceName}
```

Listing user roles

To list all the roles that a user belongs to, use the following command in wsadmin:

- Using Jython:

```
AdminTask.listAllRolesForUser("-bus busname -user userName")
```

- Using Jacl:

```
$AdminTask listAllRolesForUser {-bus busname -user userName}
```

Listing group roles

To list all the roles that a group belongs to, use the following command in wsadmin:

- Using Jython:

```
AdminTask.listAllRolesForGroup("-bus busname -group groupName")
```

- Using Jacl:

```
$AdminTask listAllRolesForGroup {-bus busname -group groupName}
```

Removing users and groups by using the wsadmin tool

Use these commands to remove a user or group. Removing a user or group withdraws the authorization permissions for all roles to which they have previously been added.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBAdminBusSecurityCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Commands

Removing a user

To remove a user from all the roles that they belong to, use the following command:

- Using Jython:

  ```
  AdminTask.removeUserFromAllRoles("-bus busname -user userName")
  ```

- Using Jacl:

  ```
  $AdminTask removeUserFromAllRoles {-bus busname -user userName}
  ```

Removing a group

To remove a group from all the roles that it belongs to, use the following command:

- Using Jython:

  ```
  AdminTask.removeGroupFromAllRoles("-bus busname -group groupName")
  ```

- Using Jacl:

  ```
  $AdminTask removeGroupFromAllRoles {-bus busname -group groupName}
  ```

After using these commands, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Removing authorization data by using the wsadmin tool

Use these commands to remove authorization data for the default roles, or for a destination or a foreign bus.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBAdminBusSecurityCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Commands

Removing authorization data for the defaults

To remove all users and groups from every role (sender, receiver and so on) in the defaults, use the following command:

- Using Jython:

```
AdminTask.removeDefaultRoles("-bus busname")
```

- Using Jacl:

```
$AdminTask removeDefaultRoles {-bus busname}
```

Removing all authorization data for a destination

To delete all authorization data for a destination, use the following command:

- Using Jython:

```
AdminTask.removeDestinationRoles("-type destinationType -bus busname
 -foreignBus foreignBusName -destination destinationName")
```

- Using Jacl:

```
AdminTask.removeDestinationRoles("-type destinationType -bus busname
 -foreignBus foreignBusName -destination destinationName")
```

Notes:

- This command deletes all authorization data for the specified destination. If the destination is a topic space, the command removes all authorization data for the virtual root and for the topics within the topic space, as well as for the topic space itself.
- You can use this command if you are deleting the destination and want to remove all associated authorization permissions, or if you want to block all access to a destination by removing all authorization permissions for it. In this second case a user might still be able to access the destination if they have been granted default authorization permissions. If you have specified default authorization permissions and you want to block all access to the destination, you must stop the destination inheriting the default permissions by using the command `setInheritDefaultsForDestinations` (see “Defining destination defaults inheritance by using the wsadmin tool” on page 2417). You should use the `removeDestinationRoles` command first, followed by the `setInheritDefaultsForDestinations` command.

Removing all authorization data for a foreign bus

To delete all authorization data for the specified foreign bus, use the following command:

- Using Jython:

```
AdminTask.removeForeignBusRoles("-bus busname
 -foreignBus foreignBusName")
```

- Using Jacl:

```
$AdminTask removeForeignBusRoles {-bus busname
 -foreignBus foreignBusName}
```

After using these commands, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

listSIBPermittedChain command

Use the listSIBPermittedChain command to list the permitted transport chains for a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The listSIBPermittedChain command lists the permitted transport chains for a selected service integration bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

Conditional parameters

None.

Optional parameters

None.

Example

The following example lists the permitted transport chains for a bus called `Bus1`.

```
AdminTask.listSIBPermittedChain('[-bus Bus1]')
```

addSIBPermittedChain command

Use the addSIBPermittedChain command to add a new transport mechanism to the list of permitted transports for a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`

- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The addSIBPermittedChain command add a new transport chain to the list of permitted transports for a selected service integration bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-chain *chainName*
The name of the transport chain you want to add to the list of permitted transports.

Conditional parameters

None.

Optional parameters

None.

Example

The following example adds the transport chain `MyTransportChain` to a bus called `Bus1`.

```
AdminTask.addSIBPermittedChain('[-bus Bus1 -chain MyTransportChain]')
```

removeSIBPermittedChain command

Use the removeSIBPermittedChain command to remove a selected transport chain from the list of permitted transport chains for a selected service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

This command is valid only when used with WebSphere Application Server Version 7.0 or later application servers. Do not use it with earlier versions.

Command-line help is provided for service integration bus commands:

- For a list of the available service integration bus security commands in Jython and a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBAdminBusSecurityCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The removeSIBPermittedChain command removes a selected transport chains from the list of permitted transport chains for a selected service integration bus.

Target object

None.

Required parameters

-bus *busName*
The name of the local bus. You can use the listSIBuses command to list the names of existing buses.

-chain *chainName*
The name of the transport chain you want to remove from the list of permitted transports.

Conditional parameters

None.

Optional parameters

None.

Example

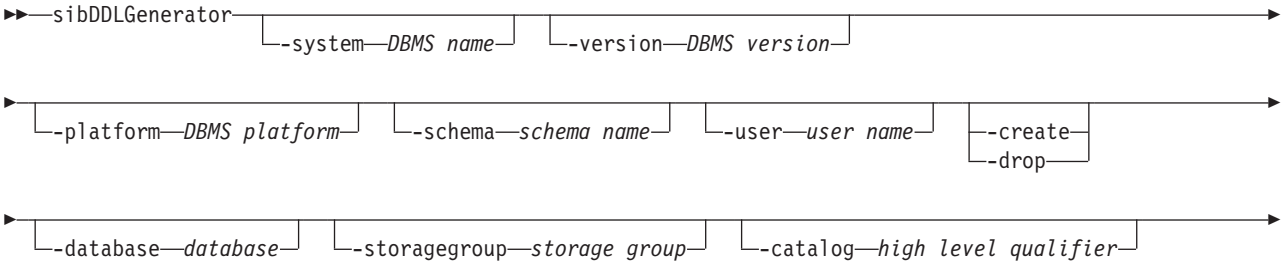
The following example removes the transport chain called `MyTransportChain` from the list of permitted transport chains for a bus called `Bus1`.

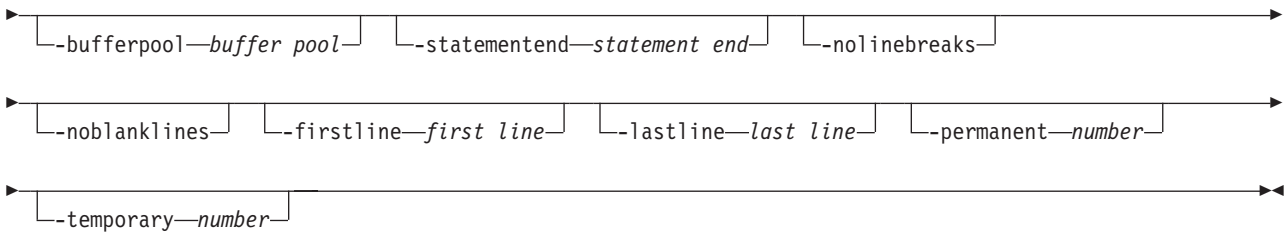
```
AdminTask.removeSIBPermittedChain('[-bus Bus1 -chain MyTransportChain]')
```

sibDDLGenerator command

A messaging engine needs data definition language (DDL) statements to create the DBMS (Database Management System) resources. These DBMS resources are generated by the sibDDLGenerator command.

Syntax





Purpose

A messaging engine needs DBMS resources, such as database tables, which it can create when starting. If your installation has a policy that only a database administrator has the authority to create database tables, use the sibDDLGenerator command to enable your database administrator to create the DBMS resources that the messaging engine needs. The sibDDLGenerator command generates the DDL statements that your database administrator can save, and later process, to create the DBMS resources that are listed in “Data store tables” on page 1967.

The command also generates DDL statements that grant the appropriate authorities to allow a messaging engine use these tables.

To access the IBM i command line, use the STRQSH command to start a Qshell session. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Note: The sibDDLGenerator command is able to generate the DDL statements for creating or dropping all of the data store tables. It is less well suited for adding more item tables to an existing data store. However, it is possible to reissue the DDL statements to create existing tables without harming the data store.

Optional parameters

Tip: The sibDDLGenerator command is case-sensitive. For example, the parameter value Oracle is not the same as the parameter value oracle.

Some parameters apply to a specific DBMS only, as indicated in the parameter list; for example:
-database [Applies only to DB2 for z/OS]

-system
Identifies the type of DBMS on which the administrator will process the DDL statements. Valid values are:

- db2
- oracle
- sqlserver
- sybase
- informix
- derby
- cloudscape

If you do not supply a **-system** parameter, the default value is derby.

-version
The version number of the DBMS identified by the **-system** parameter. The following table shows the default value that is used if you do not supply a **-version** parameter.

If you receive a CWSIS1595E or CWSIS1503E error when you run the tool, consult the notes after the table to determine whether an alternative **-version** parameter value can be used for your database version.

Table 225. Values for the **-version** *parameter. The first column of the table lists the -system parameter values. The second column of the table lists the -platform parameter values. The third column of the table lists the default values of the -version parameters. The fourth column of the table lists the other accepted values of the -version parameters.*

-system parameter value	-platform parameter value	Default value for the -version parameter	Other accepted -version parameter values
db2	zos	7.1	8.1, 9.1 (see Note 1)
db2	unix, windows	8.1	9.1 (see Note 1)
db2	iseries	5.2 (see Note 2)	5.3 (see Note 2)
oracle		8i	9i, 10g, 11g (see Note 3)
sqlserver		2000	2005
sybase		12.0	12.5, 15.0
informix		9.3	7.3, 9.4, 10.0, 11.0 (see Note 4)
derby		10.1	

Notes:

1. The DDL generated for DB2 for UNIX or Windows by using **-version** parameter values 8.1, and 9.1 is identical. The DDL generated for DB2 for z/OS by using **-version** parameter values 7.1, 8.1, and 9.1 is identical.
2. For IBM i, the DB2 database is part of the operating system, so the version number given is for the OS/400 version.
3. The DDL generated for Oracle by using **-version** parameter values 9i, 10g, and 11g is identical.
4. The DDL generated for Informix by using **-version** parameter values 9.4, 10.0, and 11.0 is identical.

-platform
The operating system platform on which the DBMS runs. Valid values are:

- iseries
- unix
- windows
- zos

-schema
The name of the schema that contains all the objects used by the messaging engine. If you do not supply a **-schema** parameter, the default value is IBMWSSIB.

-user
The name of the user ID that is used to interact with the DBMS. If you do not supply a **-user** parameter, the default value is IBMUSER.

-create | -drop
Indicates whether the DDL statements create the DBMS resources or delete them. If you do not supply either parameter, the default value is **-create**.

-database [Applies only to DB2 for z/OS]
The name of the database that is allocated for the messaging engine tables. If you do not supply a **-database** parameter, the default value is SIBDB.

-storagegroup [Applies only to DB2 for z/OS]
The name of the storage group that is allocated for the messaging engine tables.

- If you supply both the **-storagegroup** and the **-catalog** parameters, the sibDDLGenerator command includes both values in the CREATE STOGROUP statement.
- If you supply only a **-storagegroup** parameter, the sibDDLGenerator command uses the storage group name in other statements but does not create a CREATE STOGROUP statement.
- If you supply only a **-catalog** parameter, the sibDDLGenerator command displays the usage statement and then terminates.
- If you omit both parameters, the sibDDLGenerator command uses the default value SIBSG for the storage group name in other statements but does not create a CREATE STOGROUP statement.

-catalog [Applies only to DB2 for z/OS]
The name of the high level qualifier for the storage group that is allocated for the messaging engine tables. For information about defaults, refer to the **-storagegroup** parameter.

-bufferpool [Applies only to DB2 for z/OS]
The name of the buffer pool that is allocated for the messaging engine tables. If you do not supply a **-bufferpool** parameter, the default value is BP1.

-statementend
Appends *statement_end* to each DDL statement. For example, you can use ; to append a semicolon to each DDL statement. By default, the sibDDLGenerator command appends nothing to each statement.

Tip: On UNIX platforms, escape the semicolon to prevent the shell from interpreting it.

-nolinebreaks
Places each statement on a single line, with no line breaks. By default, the sibDDLGenerator command breaks statements across lines to improve readability.

-noblanklines
Omits blank lines between each statement. By default, the sibDDLGenerator command inserts a blank line between each statement to improve readability.

-firstline
Generates *first_line* as the first line of output. For example, you can use *first_line* to identify the target database. By default, the sibDDLGenerator command does not generate a first line.

-lastline
Generates *last_line* as the last line of output. For example, you can use *last_line* to invoke a command that executes the commands in the script. By default, the sibDDLGenerator command does not generate a last line.

Tip: The optional parameters that control the format of the DDL statements, for example **-statementend**, enable you to generate output that is suitable for particular scripting tools, for example the DB2 CLP. By default, the sibDDLGenerator command generates blank lines between each DDL statement but does not append a semicolon at the end of each DDL statement.

The following two optional parameters are used for spreading the data store across multiple tables:

-permanent
The number of permanent tables, with

- Default value: 1
- Minimum value: 1
- Maximum value: see **Note**

-temporary
The number of temporary tables, with

- Default value: 1
- Minimum value: 1
- Maximum value: see **Note**

Note: The maximum number of SIB*nnn* tables that can be used by a messaging engine is 32. This includes all stream, permanent and temporary tables.

Examples

- **sibDDLGenerator -system** *db2* **-version** *8.1* **-platform** *zos*

  Generates DDL statements for DB2 8.1, running on z/OS, with a default schema, user ID, database, storage group, and buffer pool.
- **sibDDLGenerator -system** *db2* **-version** *8.1* **-platform** *windows* **-statementend** *;*

  Generates DDL statements for DB2 8.1, running on Windows, with a default schema, user ID, and database. You can input the statements directly to the DB2 CLP, which requires that each statement is terminated with a semicolon.
- **sibDDLGenerator -system** *oracle* **-version** *8i* **-schema** *SIB* **-user** *fred*

  Generates DDL statements for Oracle 8i.
- **sibDDLGenerator -system** *oracle* **-schema** *SIB* **-user** *fred*

  A concise version of the preceding example.

Access role assignments for bus security resources

Icons are used in the administrative console to represent users and groups that have access roles for service integration bus resources.

Security access roles

In the administrative console, access role icons are used to represent whether a user or a group is in a particular access role. You can click an icon to add or remove selected users and groups to a particular access role for a selected resource.

An access role icon has three states:

- Access role type set.
- Access role type not set.
- Access role type inherited from group.

The following table describes how the access role icons represent these states, and how to change between them:

Table 226. Interacting with access role icons

Access role icon	Access role assignment state	User action
	Role type not set.	Click to change to role type set  .
	Role type set.	Click to change to role type not set. The icon changes to role type not set  if the user or group does not inherit access roles, or to role type inherited  if the role type does inherit access roles.
	Role type inherited from group.	Click to change to role type set  .

Table 226. Interacting with access role icons (continued)

Access role icon	Access role assignment state	User action
	Role type not set for a group. The group to which a user belongs does not have a role type.	Read only.
	Role type set for a group. The group to which a user belongs has a role type.	Read only.
	Role type not applicable.	Read only.

Message properties support for mediations

The SIMessage metadata properties enable the main data types, and are supported by JMS Message Selectors.

Syntax

The selector syntax is the same as for JMS Message Selectors. For more information, refer to the JMS specification.

Supported types

Support is provided for the following data types:

- Boolean
- Byte and byte[]
- Integer
- Long
- Float
- Double
- String

The data types listed above are supported by JMS Message Selectors with the exception of byte[].

Properties

For more information about each property, refer to “JMS_IBM properties and equivalent SI_system properties” on page 2432.

SIMessage metadata properties

The SIMessage metadata properties contain message metadata that you can use in mediation configuration selectors. You can work with these properties by using the SIMessage interface.

You can access and modify the SIMessage metadata properties using individual SIMessage get and set methods. You cannot set or access these properties by using the methods `getMessageProperty()`, `setMessageProperty()`, or `deleteMessageProperty()`.

The method `clearMessageProperty()` does not clear these properties.

Property name	Data type	Comments
SI_NextDestination	String	Name of the first destination in the Forward Routing Path.

Property name	Data type	Comments
SI_Reliability	String	Value of `getReliability.toString()`.
SI_Priority	Integer	
SI_TimeToLive	Long	
SI_Discriminator	String	
SI_ReplyPriority	Integer	
SI_ReplyReliability	String	Value of `getReliability.toString()`.
SI_ReplyTimeToLive	Long	
SI_ReplyDiscriminator	String	
SI_RedeliveredCount	Integer	
SI_MessageID	String	
SI_CorrelationID	String	
SI_Userid	String	
SI_Format	String	

JMS headers

Support is provided by the SIMessage interface and the mediation configuration selector for JMS headers properties. JMS headers properties match in the SIMessage interface in the same way as they do for the JMS API, but you can only modify properties that map to SIMessage metadata.

The method `getMessageProperty()` method supports all of these properties.

The methods `setMessageProperty()` and `deleteMessageProperty()` only support JMSType.

The method `clearMessageProperties()` only clears JMSType.

Property name	Can be matched?	Can be modified?	Data type	Comments
JMSDestination	No	Indirectly	Byte array	`getMessageProperty()` returns opaque byte array.
JMSDeliveryMode	Yes	Yes, by using `setReliability`.	String	String value, as for JMS.
JMSMessageID	Yes	Yes, by using `setMessageID`.	String	Equivalent to SI_MessageID.
JMSTimestamp	Yes	No	Long	
JMSExpiration	Yes	Indirectly, by using `setTimeToLive` or `setRemainingTimeToLive`.	Long	
JMSRedelivered	Yes	No	Boolean	
JMSPriority	Yes	Yes, by using `setPriority`.	Integer	Equivalent to SI_Priority.
JMSReplyTo	No	Indirectly	Byte array	`getMessageProperty()` returns opaque byte array.
JMSCorrelationID	Yes	Yes, by using `setCorrelationId`.	String	Equivalent to SI_CorrelationID.
JMSType	Yes	Yes, by using `setMessageProperty()`.	String	

JMSX properties

Support is provided by the SIMessage interface and the mediation configuration selector for JMSX properties. You can use the SIMessage interface to match and access supported JMSX properties defined in the JMS API. You can only use the SIMessage interface to set properties that are not defined as set by the JMS provider.

The methods `getMessageProperty()`, `setMessageProperty()` and `deleteMessageProperty()` provide access and, where supported, modification.

The method `clearMessageProperties()` does not clear properties that cannot be set.

Property name	Can be matched?	Can be modified?	Data type	Comments
JMSXUserID	Yes	Yes	String	
JMSXAppID	Yes	Yes	String	
JMSXProducerTXID	No	No		Not supported
JMSXConsumerTXID	No	No		Not supported
JMSXRcvTimestamp	No	No		Not supported
JMSXDeliveryCount	Yes	No	Integer	
JMSXState	No	No		Not supported
JMSXGroupID	Yes	Yes	String	
JMSXGroupSeq	Yes	Yes	Integer	

JMS_IBM properties and equivalent SI_system properties

Support is provided by the SIMessage interface and the mediation configuration selector for JMS_IBM properties and the equivalent SI_system properties. You can access JMS_IBM_ properties through the JMS API. Many of the values held by JMS_IBM_ properties apply to an SIMessage and have SI_ synonyms. You can access all these properties through the SIMessage interface, and can match and set many of them. You cannot set exception properties because they are controlled by the messaging engine.

You can match SI_ExceptionReason, JMS_IBM_ExceptionReason and JMS_IBM_ExceptionTimestamp. The method `clearMessageProperties()` does not clear properties that cannot be set.

Where the data types are different, the equivalent values are modified before being returned to the JMS API caller. For example with JMS_IBM_Report_XXX, the JMS_IBM_Report... and JMS_IBM_Feedback values are modified before being returned to the JMS API caller. The values used by service integration and WebSphere Application Server are different, however they are modified before being returned by the JMS API caller, so that they can then be passed to WebSphere MQ.

For information about the mapping of message fields and properties between WebSphere MQ and JMS see Mapping the message header fields and properties to and from WebSphere MQ format.

In the following table, the SIMessage API data type column indicates the data type of the property if accessed by a mediation handler, or when specifying the selectors for a mediation handler. The JMS API data type column indicates the data type of the property if accessed by a JMS application, either when specifying selectors or when using the get and set property methods:

Table 227. JMS_IBM properties and SIMessage properties. The first column of the table provides the JMS_IBM property names. The second column provides the equivalent SIMessage property names if available. The third column indicates if the properties can be matched. The fourth column indicates the state of setMessageProperty. The fifth column contains the data type of the SIMessage API property if accessed by a mediation handler or when specifying the selectors for the mediation handler. The sixth column contains the data type of the JMS API property if accessed by a JMS application.

JMS_IBM property name	Equivalent SIMessage property	Can be matched?	setMessageProperty	SIMessage API data type	JMS API data type
JMS_IBM_Format		Yes	Yes	String	String
JMS_IBM_MsgType		Yes	Yes	Integer	Integer
JMS_IBM_Feedback	SI_ReportFeedback	Yes	Yes	Integer	Integer
JMS_IBM_PutApplType		Yes	Yes	Integer	Integer
JMS_IBM_Report_Exception	SI_ReportException	Yes	Yes	Byte	Integer
JMS_IBM_Report_Expiration	SI_ReportExpiry	Yes	Yes	Byte	Integer
JMS_IBM_Report_COA	SI_ReportCOA	Yes	Yes	Byte	Integer
JMS_IBM_Report_COD	SI_ReportCOD	Yes	Yes	Byte	Integer
JMS_IBM_Report_PAN	SI_ReportPAN	Yes	Yes	Boolean	Integer
JMS_IBM_Report_NAN	SI_ReportNAN	Yes	Yes	Boolean	Integer
JMS_IBM_Report_Pass_Msg_ID	SI_ReportPassMsgID	Yes	Yes	Boolean	Integer
JMS_IBM_Report_Pass_Correl_ID	SI_ReportPassCorrelID	Yes	Yes	Boolean	Integer
JMS_IBM_Report_Discard_Msg	SI_ReportDiscardMsg	Yes	Yes	Boolean	Integer
JMS_IBM_Last_Msg_In_Group		Yes	Yes	Boolean	Boolean
JMS_IBM_PutDate		Yes	Yes	String	String
JMS_IBM_PutTime		Yes	Yes	String	String
JMS_IBM_Encoding		Yes	Yes	Integer	Integer
JMS_IBM_Character_Set		Yes	Yes	String	String
JMS_IBM_ExceptionMessage		No	No	String	String
JMS_IBM_ExceptionTimestamp	SI_ExceptionTimestamp	Yes	No	Long	Long
JMS_IBM_ExceptionReason	SI_ExceptionReason	Yes	No	Integer	Integer
JMS_IBM_ ExceptionProblemDestination	SI_Exception ProblemDestination	Yes	No	String	String
N/A	SI_ExceptionInserts	No	No	List of strings	n/a
JMS_IBM_System_MessageID	SI_SystemMessageID	Yes	No	String	String

Using the JMS_IBM Feedback property

The JMS_IBM_Feedback property identifies the type of report a message contains.

Property	Type	Values
JMS_IBM_Feedback	Integer	• REPORT_EXPIRY=3 • REPORT_EXCEPTION=4 • REPORT_COA=5 • REPORT_COD=6

If a report message is generated as a result of a message expiry, the value of the JMS_IBM_Feedback property is 3.

User properties

Support for user properties is provided by the SIMessage interface and the mediation configuration selector. The JMS API supports user properties of primitive wrapper or string types. The property name can be any valid Java identifier providing it does not have the prefix JMS. The SIMessage API also supports user properties of primitive wrapper or string types, and additionally supports byte[] and serializable types. Arbitrary serializable objects are stored as byte arrays, and are selected on as byte arrays only (using equals only).

User properties supported by the SIMessage API must have the prefix `user`. You can set and access these properties using `getMessageProperty`, `setMessageProperty` and `deleteMessageProperty`.

Interaction with JMS

Alternatively, you can set and access user properties by using `xxxUserProperty` methods. In this case, the prefix `user` must be omitted. The property name, excluding the prefix `user`, exists in the same namespace as the JMS user properties.

For example, a JMS application calls a property as follows:

```
setStringProperty("color", "green");
```

A mediation can access the property by making one of the following calls:

- `getMessageProperty("user.color");`
- `getUserProperty("color");`

Note: Mediation message selectors must contain the `user` prefix.
JMS property methods only affect user properties that have types supported by the JMS API:

- `clearProperties()` clears only those properties supported by JMS.
- `propertyExists()` returns true only when the property type is supported by JMS.
- `getPropertyNames()` includes only those properties with types supported by JMS.
- `setObjectProperty("xxxx", null);` clears a property only if it is supported by JMS.

Note that `setxxxxProperty("xxxx", value)` overrides a user property of any type when the value is non-null.

Error handling in mediations

The actions taken in the event of an error occurring during mediation processing are summarized in the following table:

Table 228. Actions and errors in mediation processing. The first column of the table lists the errors that occur during the mediation processing. The second column describes the actions to be taken when the errors occur.

Error	Action taken
Unchecked runtime exception	• The message is sent to the exception destination. • Any transaction is rolled back.
Checked message context exception	• The message is sent to the exception destination. • Any transaction is committed.
Enterprise JavaBeans (EJB) exception	• Message is eligible for re-mediation.
An error occurs in the process of calling a mediation.	• The mediation is not called. • The message is eligible for re-mediation.

Table 228. Actions and errors in mediation processing (continued). The first column of the table lists the errors that occur during the mediation processing. The second column describes the actions to be taken when the errors occur.

Error	Action taken
The mediation returns true, and the message is not well formed.	• The original pre-mediated message is sent to the exception destination. • Any transaction is committed.

WebSphere MQ naming restrictions

The naming restrictions for WebSphere MQ queues, queue managers, and queue-sharing groups are more restrictive than those that apply to equivalent objects in WebSphere Application Server. Use this information to help you administer the WebSphere Application Server objects, so that the names of these objects can be passed successfully to and from WebSphere MQ.

Rules for naming WebSphere MQ objects

- A WebSphere MQ object cannot have the same name as any other object of the same type.
- Names in WebSphere MQ are case sensitive.
- In WebSphere MQ, the names of queues can have up to 48 characters. The names of queue managers can have also have up to 48 characters, except in the case of WebSphere MQ for z/OS, where both queue manager and queue sharing group names are limited to four non-blank characters.
- The character set to use for naming all WebSphere MQ objects is as follows:
  - Uppercase A-Z
  - Lowercase a-z (but there are restrictions on the use of lowercase letters for z/OS console support)
  - Numerics 0-9
  - Period (.)
  - Forward slash (/).
  - Underscore (_)
  - Percent sign (%).
- Leading or embedded blanks are not allowed.
- Any structure to the names (for example, the use of the period or underscore) is not significant to the queue manager.

For further details, see the WebSphere MQ information center.

Mediation thread pool properties

The table below describes the default thread pool properties for the mediationsThreadPool object for a messaging engine.

Property name	Description	Data type	Comment
minimumSize	Specifies the minimum number of threads to allow in the pool.	integer	Default value is 1.
maximumSize	Specifies the maximum number of threads to allow in the pool.	integer	Default value is 5.
isGrowable	Can the pool grow beyond the maximumSize	boolean	true or false. Default is false.

Property name	Description	Data type	Comment
inactivityTimeout	Specifies the number of milliseconds of inactivity that should elapse before a thread is reclaimed. A value of 0 indicates not to wait and a negative value (less than 0) means to wait forever.	integer, units milliseconds.	Default value: 3500

Chapter 25. Administering Session Initiation Protocol (SIP) applications

This page provides a starting point for finding information about SIP applications, which are Java programs that use at least one Session Initiation Protocol (SIP) servlet written to the JSR 116 specification.

SIP is used to establish, modify, and terminate multimedia IP sessions including IP telephony, presence, and instant messaging.

Deploying SIP applications

Use the administrative console to customize your Session Initiation Protocol (SIP) application installation

About this task

When you deploy a Session Initiation Protocol (SIP) application, you can perform various tasks such as installing, starting, stopping, upgrading, and uninstalling the application.

SIP applications are installed as Java Platform, Enterprise Edition (Java EE) applications. You can deploy a SIP application from a graphical interface or from a command line.

Deploying SIP applications through the console

You can deploy a Session Initiation Protocol (SIP) application through the administrative console.

Before you begin

SIP applications are deployed as Java 2 Platform Enterprise Edition (J2EE) applications. In order to process requests, a virtual host must be defined when deploying the SIP application. If there is no virtual host defined for the configured SIP container listen port, the installed application will be inaccessible.

Procedure

1. Open the administrative console.

   In a browser, go to URL `http://`*`hostname`*`:9090/admin`, where *hostname* is the name of the host computer. Enter the appropriate login information, and click **OK**.
2. In the left frame click **Applications** > **Install New Application**.
3. Browse and select a SAR file. Specify the context root, beginning with a slash (/), in the **Context Root** field. For example, if your application is named ThisApplication, type `/ThisApplication`.
4. Click **Next** (under the **Context Root** field not beside the WebSphere Status title). If the SAR file has been assembled correctly, the screen will still have the title “Preparing for the application installation”, but the content will change. If an error message appears, check the contents of the SAR file; in particular, verify the `web.xml` file contents, and try to reload the SAR file.
5. Click **Next**. If you see a screen indicating “Application Security Warnings”, click **Continue**.
6. The **Install New Application** screen should appear with “Step 1: Select application options” highlighted. Select the options you need and click **Next**.
7. “Step 2: Map modules to servers” should appear highlighted now. You can choose the cluster or server where you want to install the application's modules.
   - If you are installing the application in a stand-alone system, click **Next**.
   - If you are installing the application in a clustered system, select **WebSphere:cell=*cellname*,cluster=*cluster_name*** in the **Clusters and Servers** field, select the check box beside the web module that you want to install, and click **Apply** and **Next**.

8. Now “Step 3: Map virtual hosts for web modules” should appear highlighted. To the right of the application name there should be a drop-down labeled **Virtual Host**.
   - If you are installing the application in a stand-alone system, set the value of the drop-down to **default_host**, and click **Next**.
   - If you are installing the application in a clustered system, set the value of the drop-down to the name of the virtual host that was chosen during setup, and click **Next**.

   Remember: You must define a virtual host for your configured SIP container listen port or else you will not be able to access the application.
9. You should now see “Step 4: Summary” highlighted. In the right panel you will see a **Summary of installation options** table that details your selected options and their values. If you need to change an option, click **Previous** to return to the section where you can make your change. Click **Finish** to install the application with your settings. The screen should display, `Application` *appname*`_sar installed successfully`, where *appname* is the name of the application.
10. Click the **Save to Master Configuration** link. A Save to Master Configuration window appears.
11. In the Save to Master Configuration window, click **Save**. The application has now been saved in the current configuration.
12. To confirm that the installation succeeded, in the left frame click **Applications** > **Enterprise Applications**. The newly installed application should appear in the list of installed applications as *appname*_sar.
13. To start the application so that it can service SIP requests, check the box beside *appname*_sar, and click **Start**. You might also want to look at the logs for a successful startup message.

Results

The application can service SIP requests now.

Deploying SIP applications through scripting

You can deploy a Session Initiation Protocol (SIP) application not only from the administrative console but also from a command line.

About this task

Note: To deploy a SIP application, the application must exist with an enterprise archive (EAR) file, a Session Initiation Protocol (SIP) module (SAR file), or a web application archive (WAR) file.

Use the wsadmin scripting tool to deploy applications from a command line.

Procedure

- Launch a scripting client.

  For more information, see AdminApp object for scripted administration.
- List applications.

  For more information, see Listing applications using the wsadmin scripting tool.
- Install stand-alone archive files.

  For more information about installation, see Installing enterprise applications using wsadmin scripting and Installation options for the AdminApp object.
- Edit application configurations.

  For more information, see Editing application configurations using the wsadmin scripting tool.
- Uninstall applications.

  For more information, see Uninstalling enterprise applications using the wsadmin scripting tool.

Administering SIP applications

Configuring the SIP container

Configure the Session Initiation Protocol (SIP) container to adjust message response times or set a custom property.

About this task

Use the administrative console to configure SIP container settings. Complete the following steps to find and configure the SIP container settings.

Procedure

1. Start WebSphere Application Server.
2. From the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
3. From **Container Settings**, expand **SIP Container Settings**, and click **SIP Container**. Select the container settings that you want to change.
   - From **General Properties**, you can configure session, message, and response time maximums. See *Session Initiation Protocol container settings* and *SIP container custom properties*.
   - From **Additional Properties**, you can define custom properties, manage transport chains or inbound channel settings, or configure the session manager.
4. After configuring the SIP container, click **Apply** to save the changes.
5. Restart WebSphere Application Server.

Results

Changes to the SIP container settings take effect after you restart WebSphere Application Server.

gotcha: As the number of SIP containers grow in a deployment, the larger the heap settings need to be on the SIP proxy server. For example, a deployment of 20 containers requires a minimum heap size of 60 MB, so a -Xmo60m parameter should be added to the Generic JVM arguments field on the Java virtual machine panel of the admin console. However, a deployment of 70 containers requires a larger value such as 200 MB (-Xmo200m). See the information center topic *Java virtual machine settings* for more information about Generic JVM arguments.

Enabling Session Initiation Protocol (SIP) flow token security

The Session Initiation Protocol (SIP) container supports client-initiated connection reuse. SIP flow token security enables you to establish communication between a server and SIP clients in situations where the SIP clients can create a connection to the server, but are not prepared to accept connections from the server.

About this task

Managing client-initiated connections in the SIP container involves generating flow tokens, as described in the SIP standard RFC 5626. When the SIP container delivers a flow token to the network, it encodes the token in a way that prevents anyone from modifying this token. When the container receives a flow token that it previously generated, it decodes the flow token and verifies its integrity.

WebSphere Application Server SIP flow token security implements the outbound SIP protocol extension, as defined in RFC 5626, with the following exceptions:

- Only TCP and TLS stream transports are supported.
- UDP flows are not reused.

- TCP keepalives are supported, but STUN keepalives are not.
- Support of this protocol extension is provided for SIP applications that act as a proxy/registrar, as described in RFC 5626, but not as a user agent, as described in this RFC.

Encoding and decoding the flow token requires a pre-defined key. The SIP container obtains this security key from your SIP container settings. Complete the following steps to configure the SIP container to support flow token security.

Procedure

1. Create a key set, if one does not already exist.

   If you already have a key set configured, you can use that key set as the key set for SIP flow token security.

   If you need to create a new key set, the scope of the key set must be at the cell level. See the topic *Creating a key set configuration* for a description of how to create a new key set.
2. Add the com.ibm.ws.sip.key.set custom property to the SIP container settings.
   a. In the administrative console, expand **Servers > Server Types > WebSphere application servers > *server_name*** to open the configuration tab for the server.
   b. From **Container settings**, expand **SIP Container settings**, and click **SIP container**.
   c. From **Additional properties**, select **Custom Properties** > **New**.
   d. On the settings page, specify `com.ibm.ws.sip.key.set` in the **Name** field, and the name of the key set to use for flow token security in the **Value** field.
   e. Click **Apply** or **OK**.
   f. Click **Save** on the console task bar to save your configuration changes.
   g. Restart the server.

Results

SIP flow token security is enabled for the SIP container.

SIP container custom properties

You can add any of the following custom properties to the configuration settings for a Session Initiation Protocol (SIP) container.

To specify custom properties for a specific SIP container, navigate to the custom properties page, and then specify a value for the custom property.

Important: The custom properties are supported as the primary method of configuration. Therefore, if a custom property is set and then you set the corresponding setting in the administrative console, the custom property value is used.

1. In the administrative console, expand **Servers > Server Types > WebSphere application servers > *server_name*** to open the configuration tab for the server.
2. From **Container settings**, expand **SIP Container settings**, and click **SIP container**.
3. From **Additional properties**, select **Custom Properties** > **New**.
4. On the settings page, type the custom property to configure in the **Name** field, and then type the value of the custom property in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

The following list of SIP container custom properties is provided with the product. These properties are not shown on the settings page for the container.

You can define the following SIP container custom properties that are provided with the product. These properties are not shown on the settings page for the container.

- “auth.int.enable” on page 2442
- “com.ibm.sip.sm.lnm.size” on page 2442
- “com.ibm.webspehere.sip.security.digest.ldap.cachecleanperiod” on page 2442
- “com.ibm.websphere.sip.security.tai.usercachecleanperiod” on page 2442
- “com.ibm.ws.sip.key.set” on page 2442
- “com.ibm.ws.sip.tai.DisableSIPBasicAuth” on page 2442
- “DigestPasswordServerClass” on page 2443
- “enable.system.headers.modify” on page 2443
- “end.of.service.replication” on page 2443
- “immediate.replication” on page 2443
- “javax.servlet.sip.ar.dar.configuration” on page 2443
- “javax.servlet.sip.ar.spi.SipApplicationRouterProvider” on page 2443
- “javax.sip.bind.retries” on page 2444
- “javax.sip.bind.retry.delay” on page 2444
- “javax.sip.detect.pre.escaped.params” on page 2444
- “javax.sip.force.connection.reuse” on page 2444
- “javax.sip.hide.message.body” on page 2445
- “javax.sip.hide.message.headers” on page 2445
- “javax.sip.hide.request.uri” on page 2445
- “javax.sip.OUTBOUND_PROXY” on page 2445
- “javax.sip.PATH_MTU” on page 2446
- “javax.sip.stat.report.interval” on page 2446
- “javax.sip.trace.msg.in” on page 2446
- “javax.sip.trace.msg.out” on page 2446
- “javax.sip.transaction.invite.auto100” on page 2446
- “javax.sip.transaction.timer.a” on page 2447
- “javax.sip.transaction.timer.b” on page 2447
- “javax.sip.transaction.timer.cancel” on page 2447
- “javax.sip.transaction.timer.d” on page 2447
- “javax.sip.transaction.timer.e” on page 2448
- “javax.sip.transaction.timer.f” on page 2448
- “javax.sip.transaction.timer.g” on page 2448
- “javax.sip.transaction.timer.h” on page 2449
- “javax.sip.transaction.timer.i” on page 2449
- “javax.sip.transaction.timer.invite.server” on page 2449
- “javax.sip.transaction.timer.j” on page 2449
- “javax.sip.transaction.timer.k” on page 2450
- “javax.sip.transaction.timer.non.invite.server” on page 2450
- “javax.sip.transaction.timer.t1” on page 2450
- “javax.sip.transaction.timer.t2” on page 2450
- “javax.sip.transaction.timer.t4” on page 2451
- “on.outgoing.message.replication” on page 2451
- “pws_atr_name” on page 2451

- “replicate.with.confirmed.dialog.only” on page 2451
- “sip.container.heartbeat.enabled” on page 2451
- “sip.jsr289.parse.address” on page 2452
- “SIP_RFC3263_nameserver” on page 2452
- “thread.message.queue.max.size” on page 2453
- “weight.overload.watermark” on page 2453

auth.int.enable:

Specifies the **auth-int** quality of protection (QOP) for digest authentication. Digest authentication defines two types of QOP: **auth** and **auth-int**. By default, **auth** is used. When this custom property is set to True, the highest level of protection is used, which is the **auth-int** QOP.

Data type	String
Default	False

com.ibm.sip.sm.lnm.size:

Specifies the number of logical names in the application server. Each SIP object that can be replicated, such as a SIP session, is associated with a logical name. All objects with the same logical name are replicated to the same back-up container. The proxy can route messages to the correct container using the logical name found in the message. The value must be greater than 1.

Data type	String
Default	10

com.ibm.webspehere.sip.security.digest.ldap.cachecleanperiod:

Specifies the clean Lightweight Directory Access Protocol (LDAP) cache period in minutes.

Data type	String
Default	120

com.ibm.websphere.sip.security.tai.usercachecleanperiod:

Specifies the clean security subject cache period in minutes.

Data type	String
Default	15

com.ibm.ws.sip.key.set:

Specifies the key to use for SIP flow token security. When a value is specified for this property, SIP flow token security is automatically enabled.

Data type	String
Default	There is no default value

com.ibm.ws.sip.tai.DisableSIPBasicAuth:

Specifies whether to allow basic authentication for SIP.

Data type String
Default False

DigestPasswordServerClass:

Specifies types of user registries that are supported, except LDAP. To configure DigestTAI without the LDAP user registry, complete the following steps.

1. Create a class that implements this interface: com.ibm.ws.sip.security.digest.DigestPasswordServer
2. Add the following property to the SIP container custom property:

   Default LdapPasswordServer
3. Ensure that all users declared by the impl class are declared in the user registry configured for the product security.

Data type String
Default impl

enable.system.headers.modify:

Specifies whether the application has access to headers that are otherwise restricted.

Data type String
Default False

end.of.service.replication: Specifies whether changes are buffered until the thread for a siplet is about to end. If the value is set to `true`, then each change is buffered until the thread for the siplet is about to end.

Data type Boolean
Default true

immediate.replication: Specifies whether each change is immediately sent to the Data Replication Service. When this property is set to `true`, when replication is issued from a non-SIP container thread, the replication is immediately performed on the calling thread. When this property is set to `false`, the changes are buffered, and replication does not occur until all changes are made.

Setting this property to true might have a negative impact on performance.

Data type Boolean
Default false

javax.servlet.sip.ar.dar.configuration:

Specifies the location of the default application router (DAR) properties file. The properties file defines the order in which the application router sends SIP requests to applications as described in Appendix C of the JSR 289 specification.

Data type String
Default Null

javax.servlet.sip.ar.spi.SipApplicationRouterProvider:

Specifies the custom application router implementation fully qualified class name as described in section 15.4.2 of the JSR 289 specification. The custom application router implementation class defines the order in which the application router sends SIP requests to applications.

Data type	String
Default	Null

javax.sip.bind.retries:

Specifies the amount of time, in milliseconds, between attempts to start the SIP channel if the SIP port is busy with another process during server startup.

Data type	String
Default	60

javax.sip.bind.retry.delay:

Specifies the delay, in milliseconds, between attempts to start the SIP channel if the SIP port is busy with another process during server startup.

Data type	String
Default	5000

javax.sip.detect.pre.escaped.params:

Specifies whether to prevent the container from re-escaping Uniform Resource Identifier (URI) parameters that were pre-escaped by the application.

Enabling this property provides the application with more control over escaping URI parameters, when calling the **javax.servlet.sip.SipFactory.createURI()** and the **javax.servlet.sip.SipURI.setParameter()** parameters.

By default, the container only escapes characters that it must encode according to the RFC 3261 25.1 specification. In some cases, however, escaping additional characters might be required. Due to a limitation in the JSR 116 (5.2.1) specification, the application cannot perform its own escaping. Because of this limitation, attempts by the application to encode URI parameters causes the container to re-encode the percent sign. If the value of this property is set to true, the container cannot re-encode the percent sign.

Setting the value to true is not in compliance with the JSR 116 (5.2.1) specification, but provides the application with greater control over URI parameter escaping. APAR PK37192 describes the problem and the workaround.

Data type	String
Default	False

javax.sip.force.connection.reuse:

Specifies whether to force reuse of inbound connections for outbound requests. This custom property is only relevant for stream transports, such as Transmission Control Protocol (TCP) and Transport Layer Security (TLS). Disabling this property causes the container to create a separate connection for outbound requests, even if an existing connection is already established to the same peer address. The connection is automatically reused if the top Via header in the inbound request contains an alias parameter. (http://www.ietf.org/internet-drafts/draft-ietf-sip-connect-reuse-07.txt)

Data type	String
Default	False

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.hide.message.body:

Specifies to hide message content in logs. Set the value of this property to `true` to remove the message body text from SIP messages printed in the log files. This property only affects the representation of the messages in log files.

Data type	String
Default	False

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.hide.message.headers:

Specifies to hide the specified message header field names in log files. The value of this property is a comma-separated list of header field names that you want removed from SIP messages printed in the log files. This property only affects the representation of the messages in log files.

Data type	String
Default	None

javax.sip.hide.request.uri:

Specifies to hide request URIs in log files. Set the value of this property to `true` to remove request URIs from SIP messages printed in the log files. This property only affects the representation of the messages in log files.

Data type	Boolean
Default	False

javax.sip.OUTBOUND_PROXY:

Specifies the fixed address for routing all outbound SIP messages. The format is *address:port/transport*, such as `1.2.3.4:5065/tcp`.

Note: Do not use this property if the container is fronted by an application server SIP proxy.

Data type	String
Default	null

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.PATH_MTU:

Specifies the maximum transmission unit, in bytes, for outbound User Datagram Protocol (UDP) requests. The SIP stack measures the size of a request before sending it out on the UDP channel. If the request is larger than the value specified for **PATH_MTU-200** (1300 bytes by default), then the transport is switched from UDP to TCP before transmission. Increase this value to send larger requests over the UDP channel; however, messages might be truncated or dropped. See the RFC 3261-18.1.1 specification for details.

Data type	String
Default	1500

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.stat.report.interval:

Specifies the amount of time, in milliseconds, for reporting dispatch and timer statistics to a system.out file. A value of zero indicates no report.

Data type	String
Default	0

javax.sip.trace.msg.in:

Specifies whether to print incoming messages to a system.out file.

Data type	String
Default	False

javax.sip.trace.msg.out:

Specifies whether to print outbound messages to a system.out file.

Data type	String
Default	False

javax.sip.transaction.invite.auto100:

Specifies whether to automatically reply to invite requests with a `100 Trying` response. Disabling this property might increase the number of invite retransmissions.

Data type	String
Default	True

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.a:

Specifies, for UDP only, the amount of time, in milliseconds, before retransmitting invite requests for timer A for the RFC 3261 specification. This property is relevant for the invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.b:

Specifies the amount of time, in milliseconds, for the invite client transaction timeout timer (timer B) for the RFC 3261 specification.

Data type	String
Default	64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.cancel:

Specifies the amount of timer, in milliseconds, for the timer to keep the cancelled client transaction in the proceeding state before completing the cancelled transaction for the RFC 3261 9.1 specification. This property is relevant for the invite client transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

javax.sip.transaction.timer.d:

Specifies the wait time, in milliseconds, before retransmission of the invite response for timer D for the RFC 3261 specification. This property is relevant for the invite client transaction.

Data type	String
Default	32000

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.e:

Specifies, for UDP only, the amount of time, in milliseconds, before the retransmission of the initial non-invite request for timer E for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.f:

Specifies the amount of time, in milliseconds, for the non-invite transaction timeout timer (timer F) for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.g:

Specifies the amount of time, in milliseconds, before retransmission of an initial invite response for timer G for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.h:

Specifies the amount of time, in milliseconds, to wait for an acknowledgement (ACK) receipt for timer H for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.i:

Specifies the amount of time in milliseconds to wait for an ACK retransmission for timer I for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	javax.sip.transaction.timer.t4

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.invite.server:

Specifies the amount of time, in milliseconds, for the timer to keep the invite server transaction in the complete state. This timer is not defined in the RFC specification.

To avoid creating a new server transaction when a client retransmits an invite request, keep the completed server transaction for a period of time before removing invite retransmissions. This timer is started when the transaction changes to the terminated state. When the timer completes, the transaction is removed.

Data type	String
Default	32000

javax.sip.transaction.timer.j:

Specifies the amount of time in milliseconds to wait for non-invite request retransmission for timer J for the RFC 3261 specification. This property is relevant for the non-invite server transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.k:

Specifies the amount of time, in milliseconds, to wait for non-INVITE response retransmissions for timer K for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t4

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.non.invite.server:

Specifies the amount of time, in milliseconds, for an Application Programming Interface (API) timer for the application to respond to a non-invite request. This property is relevant for non-invite server transactions.

This timer is not defined in the RFC specification. This property is needed to stop the transaction if the application does not generate a final response to the request. The timer starts when the request arrives in the stack and stops when a response is generated by the application. If no response is generated before the timer stops, then the transaction completes.

Data type	String
Default	34000

javax.sip.transaction.timer.t1:

Specifies the amount of time, in milliseconds, for a network round trip delay for timer T1 for the RFC 3261 specification. The value is used as a base for calculating some timers and is relevant for all types of transactions, such as client, server, invite, and non-invite transactions.

Data type	String
Default	500

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.t2:

Specifies the maximum time in milliseconds before retransmitting non-invite requests and invite responses for timer T2 for the RFC 3261 specification.

Data type	String
Default	4000

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.t4:

Specifies the maximum amount of time, in milliseconds, for a message to remain in the network. This value is used as a base for calculating other timers for timer T4 for the RFC 3261 specification.

Data type	String
Default	5000

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

on.outgoing.message.replication: Specifies whether changes are buffered until a siplet issues a request.send() or response.send() call. If the value is set to `true`, then each change is buffered until a siplet issues a request.send() or response.send() call.

Data type	Boolean
Default	false

pws_atr_name:

Specifies the LDAP attribute name that stores the user password.

Data type	String
Default	userpassword

replicate.with.confirmed.dialog.only:

Specifies whether to replicate the application session, even when no dialogs are confirmed. If the value is set to `false`, then the application session is replicated immediately after the session is created. Otherwise, the application session is only replicated when an associated dialog is confirmed.

Data type	String
Default	False

sip.container.heartbeat.enabled:

Specifies whether or not SIP network outage detection is enabled for the SIP container. SIP network outage detection allows the SIP proxy to send keepalive messages to the SIP container if the value of this property is set to `true`.

If the value is set to `false` for the SIP container, then this property has no effect on the SIP proxy. However, if the value is set to `true` for the SIP container, the value should also be set to `true` for the SIP proxy to ensure that keepalive messages are answered at the SIP container and not presented to the application.

Data type	String
Default	true

sip.jsr289.parse.address:

Specifies to use the SIP Servlet Specification 1.1, JSR 289 required format for createRequest() and createAddress() methods.

Note: The JSR 289 API requires that for any SIP URI that contains address parameters, you must enclose the SIP URI in angle brackets. The default behavior of the sip.jsr289.parse.address property is compliant with JSR 289 and correctly parses the address parameter as if it belongs to the SIP address. For example, when the property is set to `false`, the SIP address, `sip:fred@acme.com;param1=1`, is converted to `<sip:fred@acme.com;param1=1>`. When the property is set to `true`, the SIP address `sip:fred@acme.com;param1=1`, is converted to `<sip:fred@acme.com;>param1=1`.

Data type	String
Default	True

SIP_RFC3263_nameserver:

Specifies whether to allow a SIP URI to be resolved through Domain Name System (DNS) into the IP address, port, and transport protocol of the next hop.

The value of the property is a string containing one or two address and port tuples, where two tuples are separated by a space. The following examples specify a one address and port tuple or a two address and port tuple.

dottedDecimalAddress@.port

hostname.domain@port

IPV6address@port

The following example values represent a single tuple.

- 1.2.3.4@53
- example.com@53
- a:b:c::d@53

The following example values represent two tuples separated by a space.

- 1.2.3.4@53 example.com@53
- a:b:c::d@53 9.32.211.14@53

Data type	String
Default	null

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

thread.message.queue.max.size:

Specifies the maximum number of events allowed in the container threads queue. When this number is exceeded, the proxy server is notified that the container is overloaded and requests for new sessions are not accepted. Instead, the container returns an error message that indicates that the container is temporarily unavailable.

This value represents the total number of messages for all queues and reflects the state of the CPU. When the CPU approaches 100%, the maximum value for this custom property is reached quickly. Configure your system to limit the queue size and prevent the queue from reaching this threshold.

Data type	String
Default	1000

weight.overload.watermark:

Specifies the threshold value for the internal weight calculated by the container. When the container calculates the internal weight to be higher than the value specified, an overloaded container becomes available for service again.

This custom property represents a percentage of the maximum internal weight, such as 30 percent when the default value is set. When the high-water mark, or maximum threshold, is exceeded, the container waits until the weight drops below the maximum weight. This value cannot exceed `10`.

Data type	String
Default	3

Using DNS procedures to locate SIP servers

The Session Initiation Protocol (SIP) can use Domain Name Server (DNS) procedures for a client to resolve a SIP Uniform Resource Identifier (URI).

About this task

WebSphere Application Server provides support for the RFC 3263 standard. This allows a SIP URI to be resolved through DNS into the IP address, port, and transport protocol of the next hop to contact.

Note: SIP does not support use of DNS procedures for a server to send a response to a back-up client if the primary client fails.

Complete these steps to configure WebSphere Application Server to support the RFC 3263 standard.

Procedure

1. Start WebSphere Application Server.
2. From the administrative console, expand **Servers**, and click **Application servers** > ***serverName***.
3. Under **General Properties**, check the **Enable locating SIP servers using DNS NAPTR records** checkbox, then fill in the **Primary DNS server name** and **Secondary DNS server name** fields.
4. Click **Apply** to save your changes.
5. Restart WebSphere Application Server.

What to do next

You must configure your DNS server in order for RFC 3263 support to work for the SIP container. The following example is a BIND db file for configuring RFC 3263 support on a DNS server.

```
; Copyright (C) 2004  Internet Systems Consortium, Inc. ("ISC")
; Copyright (C) 2001  Internet Software Consortium.
;
; Permission to use, copy, modify, and distribute this software for any
; purpose with or without fee is hereby granted, provided that the above
; copyright notice and this permission notice appear in all copies.
;
; THE SOFTWARE IS PROVIDED "AS IS" AND ISC DISCLAIMS ALL WARRANTIES WITH
; REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
; AND FITNESS.  IN NO EVENT SHALL ISC BE LIABLE FOR ANY SPECIAL, DIRECT,
; INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM
; LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE
; OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
; PERFORMANCE OF THIS SOFTWARE.

; $Id: include.db,v 1.2.206.1 2004/03/06 10:22:13 marka Exp $

; Test $INCLUDE current domain name and origin semantics

example.com. 43200 IN SOA ns.example.com. email.example.com. ( 2003032001 10800 3600 604800 86400 )
;
example.com.             43200 IN NS      ns.example.com.
;
ns.example.com.          43200 IN A       10.0.0.20
sipserver1.example.com.  43200 IN A       10.0.0.21
sipserver2.example.com.  43200 IN A       10.0.0.22
sipserver3.example.com.  43200 IN A       10.0.0.23
;
router.example.com.        43200  IN CNAME sipserver3
;
sipserver1.example.com.  43200 IN AAAA    fec0:0:0:0:0:0:0:abcd
sipserver2.example.com.  43200 IN AAAA    fec0:0:0:0:0:0:0:abba
;
_sip._udp.example.com.   43200 IN SRV 2 0 5060  sipserver1.example.com.
_sip._udp.example.com.   43200 IN SRV 2 0 5060  sipserver2.example.com.
_sip._tcp.example.com.   43200 IN SRV 1 4 5060  sipserver1.example.com.
_sip._tcp.example.com.   43200 IN SRV 1 2 5060  sipserver2.example.com.
_sips._tcp.example.com.  43200 IN SRV 0 1 5061  sipserver1.example.com.
_sips._tcp.example.com.  43200 IN SRV 0 0 5061  sipserver2.example.com.
;
example.com. 43200 IN NAPTR 0 0 "s" "SIPS+D2T" "" _sips._tcp.example.com.
example.com. 43200 IN NAPTR 1 0 "s" "SIP+D2T"  "" _sip._tcp.example.com.
example.com. 43200 IN NAPTR 2 0 "s" "SIP+D2U"  "" _sip._udp.example.com.
```

SIP container settings

Use this page to configure the SIP container settings for Session Initiation Protocol (SIP).

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **SIP container settings** > **SIP container**.

Maximum application sessions:

Specifies the maximum number of SIP application sessions that the container manages. When the maximum is reached, no new SIP conversations are started. When the maximum is exceeded in a clustered environment, the server does not forward new dialogs until the number of application sessions no longer exceeds the maximum.

Application sessions are typically created by new incoming calls, but can also be created by other events. The application session count does not impact failover, but applies only to new sessions that are created as a result of incoming calls.

When application sessions are transferred from one application server to another due to failover, the active application server inherits the sessions created on the failed server. In addition, the servlet might create a new application session in the SIP container by calling SipFactory.createApplicationSession().

New application sessions created for events other than starting SIP conversations are not controlled by this setting. But all new application sessions are included when computing the maximum number of application sessions allowed. Thus, all active application sessions, including those not related to starting SIP conversations, can cause the maximum to be exceeded.

Data type	Integer
Default	120000 (recommended)
Range	1 <= n <= java.lang.Integer.MAX_VALUE

Maximum messages per averaging period:

Specifies the maximum amount of SIP messages processed per averaging period. The averaging period is the period of time during which the average number of messages received by the container is calculated.

This average is used to determine the load for the container and to determine if the number of messages is approaching the maximum. When the maximum is exceeded, the stand-alone server or the proxy server continues to handle all in-dialog messages. Other non-dialog requests are rejected. When a container is in an overloaded state, the proxy server returns a 503 error.

Data type	Integer
Default	5000 (recommended)
Range	1 <= n <= java.lang.Integer.MAX_VALUE

Maximum dispatch queue size:

Specifies the size of the internal dispatch queue. When the maximum queue size threshold is reached, the container queue becomes overloaded and begins to drop requests for new sessions. In this case, the container does not report its overloaded state to the proxy server.

Configure your system to limit the queue size to prevent the queue from reaching this threshold. If the internal queue reaches the overloaded state, incoming UDP packets are dropped until the queue is no longer in an overloaded state. Limiting the queue size enables better recovery if the CPU is used by other processes or threads and prevents the container from reaching out-of-memory conditions. When the value is set to `0`, the queue size is unlimited.

Data type	Integer
Default	3200 (recommended)
Range	0 <= n <= java.lang.Integer.MAX_VALUE

Maximum response time:

Specifies the maximum response time, in milliseconds, for an application. When the amount of time is exceeded, the container notifies the clustering framework that it is unavailable. You can disable this feature in the administrative console by deselecting the check box and specifying a value of `0`.

Use the maximum SIP response time setting cautiously because the calculated response time does not match the behavior of all applications. For requests, such as INVITE requests, where the responses are generated as a result of a user interaction, the calculated response time is extensive. However, the extensive response time is not caused by a delay in the SIP container. Therefore, you should not calculate the response time as a load factor. The recommended applications for effective calculation of response time are applications that respond immediately without a user interaction. The subscribe and register applications are relevant examples.

Data type	Integer
Default	0
Range	1 <= n <= java.lang.Integer.MAX_VALUE

Averaging period in milliseconds:

Specifies the amount of time, in milliseconds, to use for calculating the maximum messages per averaging period. This setting is the sliding window during which the SIP container counts the number of messages sent to the container.

Data type	Integer
Default	1000 (recommended)
Range	1000 <= n <= java.lang.Integer.MAX_VALUE

Statistic update rate:

Specifies control over the interval for which the container calculates averages and publishes statistics to Performance Monitoring Infrastructure.

Data type	Integer
Default	10000 (recommended)
Range	1000 <= n <= java.lang.Integer.MAX_VALUE

Locating SIP servers using DNS:

Specifies whether to enable locating SIP servers using DNS (Directory Name Service).

A SIP Uniform Resource Identifier (URI) can be resolved through DNS into the Internet Protocol (IP) address, port, and transport protocol of the next hop.

The value for the **Primary DNS server name** or **Secondary DNS server name** fields is a string containing one address and port tuple. The following examples specify an address and port tuple.

Note: The container contacts the primary DNS server first; however, if the primary DNS server is unavailable, then the container contacts the secondary DNS server. If the secondary DNS server remains responsive, the container does not attempt to contact the primary DNS server.

dottedDecimalAddress@.port

hostname.domain@port

IPV6address@port

Data type	Boolean
Default	False

`Primary DNS server name`

Specifies an IP address and port tuple for the primary DNS server. If this server is not available, then the container sends a response to the secondary DNS server.

Data type String
Default empty string.

Secondary DNS server name
Specifies an IP address and port tuple for the secondary DNS server.

Data type String
Default empty string.

Thread pool:

Specifies the thread pool to use for the SIP container.

If you do not choose a thread pool, the SIP container creates a new thread pool with maximum size of 15.

SIP stack settings

Use this page to configure values for the Session Initiation Protocol (SIP) stack settings that are different from those specified in RFC 3261. The default values are the same as those specified for RFC 3261.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **SIP container settings** > **SIP container** > **SIP stack**.

Automatically reply to INVITE with "100 Trying" response:

Specifies whether to automatically reply to INVITE requests with a 100 Trying response. Disabling this setting might increase the number of INVITE retransmissions.

Data type Boolean
Default True

Hide message body:

Specifies whether to hide message content in log files. Set the value to `True` to remove the message body text from SIP messages printed in the log files. This setting only affects the representation of the messages in log files.

Data type Boolean
Default False

Enable outbound connection timeout:

Specifies whether to enable the outbound connection timeout.

Data type Boolean
Default False

Outbound connection timeout:

Specifies the amount of time, in milliseconds, for creating outbound connections. This setting is relevant only for stream transports, such as Transmission Control Protocol (TCP) and Transport Layer Security (TLS). The default value of `0` provides an infinite amount of time.

Data type Integer

Default 0

Maximum transmission unit:

Specifies the maximum transmission unit, in bytes, for outbound User Datagram Protocol (UDP) requests. The SIP stack measures the size of a request before sending it out on the UDP channel. If the request is higher than the value specified for PATH_MTU-200, 1300 bytes by default, then the transport is switched from UDP to TCP before transmission.

Increase this value to send larger requests over the UDP channel; however, messages might be truncated or discarded. See the RFC 3261-18.1.1 specification for details.

Data type Integer
Default 1500

Outbound proxy:

Specifies the fixed address for routing all outbound SIP messages. The format is address:port/transport, such as 1.2.3.4:5065/tcp.

Best Practice: Do not use this setting if the container is fronted by an application server SIP proxy.

Data type String
Default empty string

SIP timers settings

Use this page to set values for the Session Initiation Protocol (SIP) timers that are different from those specified in RFC 3261. SIP timers provide a mechanism for session expiration.The default values are the same as those specified for RFC 3261.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **SIP container settings** > **SIP container** > **SIP stack** > **SIP timers**.

The following SIP timers can be configured from the administrative console.

T1:

Specifies the amount of time, in milliseconds, for a network round trip delay for timer T1 for the RFC 3261 specification. The value is used as a base for calculating some timers and is relevant for all types of transactions, such as client, server, invite, and non-invite transactions.

Data type Integer
Default 500

A Specifies, for UDP only, the amount of time, in milliseconds, before retransmitting invite requests for timer A for the RFC 3261 specification. This setting is relevant for the invite client transaction.

Data type Integer
Default T1

B Specifies the amount of time, in milliseconds, for the invite client transaction timeout timer (timer B) for the RFC 3261 specification.

Data type Integer

Default T1*64

E Specifies, for UDP only, the amount of time, in milliseconds, before the retransmission of the initial non-invite request for timer E for the RFC 3261 specification. This setting is relevant for the non-invite client transaction.

Data type Integer
Default T1

F Specifies the amount of time, in milliseconds, for the non-invite transaction timeout timer (timer F) for the RFC 3261 specification. This setting is relevant for the non-invite client transaction.

Data type Integer
Default T1*64

G Specifies the amount of time, in milliseconds, before retransmission of an initial invite response for timer G for the RFC 3261 specification. This setting is relevant for the invite server transaction.

Data type Integer
Default T1

H Specifies the amount of time, in milliseconds, to wait for an acknowledgement (ACK) receipt for timer H for the RFC 3261 specification. This setting is relevant for the invite server transaction.

Data type Integer
Default T1*64

J Specifies the amount of time, in milliseconds, to wait for non-invite request retransmission for timer J for the RFC 3261 specification. This setting is relevant for the non-invite server transaction.

Data type Integer
Default T1*64

T2:

Specifies the maximum amount of time, in milliseconds, before retransmitting non-invite requests and invite responses for timer T2 for the RFC 3261 specification.

Data type Integer
Default 4000

T4:

Specifies the maximum amount of time, in milliseconds, for a message to remain in the network. This value is used as a base for calculating other timers for timer T4 for the RFC 3261 specification.

Data type Integer
Default 5000

I Specifies the amount of time, in milliseconds, to wait for an ACK retransmission for timer I for the RFC 3261 specification. This setting is relevant for the invite server transaction.

Data type Integer
Default T4

K Specifies the amount of time, in milliseconds, to wait for non-INVITE response retransmissions for timer K for the RFC 3261 specification. This setting is relevant for the non-invite client transaction.

Data type	Integer
Default	T4

D:

Specifies the amount of time to wait, in milliseconds, before retransmission of the invite response for timer D for the RFC 3261 specification. This setting is relevant for the invite client transaction.

Data type	Integer
Default	32000

Configuring SIP timers

You can configure SIP timers to set values for the Session Initiation Protocol (SIP) timers that are different from default values specified in RFC 3261. SIP timers provide a mechanism for session expiration. The default values are the same as those specified for RFC 3261.

Before you begin

About this task

You can set values for the SIP timers from the administrative console.

Note: If you have already used the custom properties to specify a value for a SIP timer, the custom property value is the primary value. Therefore, the SIP timer value specified from the administrative console is not used.

Procedure

1. From the administrative console, click **Servers** > ***server_name*** > **SIP container** > **SIP stack** > **SIP timers**.
2. Specify a value for a timer by clicking the Use Custom Value check box beside the timer.
3. Specify a value for the timer in the Value column.
4. Click **OK**.
5. Restart the application server.

Results

The container uses the times specified in the administrative console and not the RFC defaults.

SIP timer summary:

Request for Comments (RFC) 3261, “SIP: Session Initiation Protocol,” specifies various timers that SIP uses.

Table 229 on page 2461 summarizes for each SIP timer the default value, the section of RFC 3261 that describes the timer, and the meaning of the timer.

Table 229. Summary of SIP timers.

This table lists a summary of SIP timers.

Timer	Default value	Section	Meaning
T1	500 ms	17.1.1.1	Round-trip time (RTT) estimate
T2	4 sec.	17.1.2.2	Maximum retransmission interval for non-INVITE requests and INVITE responses
T4	5 sec.	17.1.2.2	Maximum duration that a message can remain in the network
Timer A	initially T1	17.1.1.2	INVITE request retransmission interval, for UDP only
Timer B	64*T1	17.1.1.2	INVITE transaction timeout timer
Timer D	> 32 sec. for UDP	17.1.1.2	Wait time for response retransmissions
	0 sec. for TCP and SCTP		
Timer E	initially T1	17.1.2.2	Non-INVITE request retransmission interval, UDP only
Timer F	64*T1	17.1.2.2	Non-INVITE transaction timeout timer
Timer G	initially T1	17.2.1	INVITE response retransmission interval
Timer H	64*T1	17.2.1	Wait time for ACK receipt
Timer I	T4 for UDP	17.2.1	Wait time for ACK retransmissions
	0 sec. for TCP and SCTP		
Timer J	64*T1 for UDP	17.2.2	Wait time for retransmissions of non-INVITE requests
	0 sec. for TCP and SCTP		
Timer K	T4 for UDP	17.1.2.2	Wait time for response retransmissions
	0 sec. for TCP and SCTP		

Performing controlled failover of SIP applications

You can take an active application out of the loop via a controlled failover.

Before you begin

SipContainerMBean is used to initiate a server quiesce through wsadmin (command line interface). This MBean is used to set the container's weight to 0, which prevents new messages from being routed to it.

WebSphere Application Server PMI is used to monitor the server's active sessions. The remaining active sessions can be watched by enabling a counter on the server being quiesced. The server can be shut down once the number of active sessions reaches an acceptable level. A script can be written to monitor active sessions and shut down the server when an acceptable threshold is achieved.

About this task

Quiescing a single server

1. On an ND machine, start the wsadmin utility:
   a. Go to `<nd_installation_path>/bin`
   b. Run the command: ./setupCmdLine.sh
   c. Run the command: ./wsadmin.sh
   d. Verify that received: wsadmin>

2. Run the command: set scBean [$AdminControl queryNames type=SipContainerMBean,process=<server name>,*]
3. Run the command: $AdminControl invoke $scBean quiesce true

From the admin console, command line or scripts

Use the following commands to stop application servers from the admin console, command line or scripts:

- **Stop:** Quiesces the application server. The sessions will failover to another server. It is important to manually quiesce the server on shutdown.
- **Immediate Stop:** Stops the server, but bypasses the normal server quiesce process that supports in-flight requests to complete before shutting down the entire server process. This shutdown mode is faster than the normal server stop processing, but some application clients can receive exceptions.
- **Terminate:** Deletes the application server process. Use this if immediate stop fails to stop the server.

Configuring SIP application routers

Use the Session Initiation Protocol (SIP) application router to select the order in which SIP applications are triggered. When configuring a SIP application router, you can either use the default application router or create a custom application router.

About this task

The SIP container provides an application router component called the Default Application Router (DAR). The DAR component uses a configuration text file, similar to a Java properties file, that defines the order in which the application router sends SIP requests to applications.

Restriction: WebSphere Application Server has a default way of sorting the order of SIP applications invocation using the Startup behavior settings. The sorting order is based on the application weight. This weighting policy only applies if you do not specify a DAR configuration file, or if a custom application router has not been associated with the server or cluster.

You can either use the DAR or a custom application router to perform application routing, as described in the procedure.

Use the following procedure to select the best method to implement the SIP application router for your configuration.

Procedure

- Use the DAR component with a DAR configuration file.
  1. In the administrative console, click **Environment** > **SIP application routers**. The table displays a list of available application routers, including the DAR component.
  2. Click the **DefaultSIPApplicationRouter** link.
  3. View the list of server and cluster targets that are associated with the application router in the **Targets** table.
  4. Targets may or may not be available. To change the target of an application router, go back to the SIP application routers panel, click on a router name, and check to see if a target is listed. If a target is listed, select a target and then click the **Move to Application Router** button. The drop-down menu lets you select another application router.
  5. For the **DefaultSIPApplicationRouter**, click a target link name to set the application routing configuration for the target.
  6. Click **Advanced application routing rules (DAR configuration)** to use a DAR configuration file, and click **Apply**.
  7. Click **Configure routing rules** to view or edit the routing rules.

8. If you have an existing DAR configuration file, you can click the **Import...** button to upload the new DAR configuration file. The **File Import** window is displayed, which allows you to browse to the file and upload it; then click the **Import** button.
9. Use the **New**, **Delete**, **Move up**, and **Move down** buttons on the **DAR Configuration File** page to create and modify routing rules.
10. Click **Save** directly to the master configuration and then restart server or cluster to pick up the changes.

- Use the DAR component with manual application ordering.
  1. In the administrative console, click **Environment** > **SIP application routers**. The table displays a list of available application routers, including the DAR component.
  2. Click the **DefaultSIPApplicationRouter** link.
  3. View the list of server and cluster targets that are associated with the application router in the Targets table.
  4. Targets may or may not be available. To change the target of an application router, go back to the SIP application routers panel, click on a router name, and check to see if a target is listed. If a target is listed, select a target and then click the **Move to Application Router** button. The drop-down menu lets you select another application router.
  5. For the **DefaultSIPApplicationRouter**, click a target link name to set the application routing configuration for the target.
  6. Click **Basic application startup order** to use the application order from the target; then click **Apply**.
  7. Click **Configure application startup order** to view the application startup order weights for the applications on this target.
  8. Enter a numerical value in the **Startup order weight** column for the application. The startup order weight determines the order in which the SIP application router sends SIP requests to applications. These values also determine the startup order of applications after a server restart. Applications with lower startup values start first.

     Restriction: If there are two or more SIP applications bundled inside one enterprise archive (EAR) application file, the bundled SIP applications will have the same weight. If more complex routing rules are needed, a different application router method must be used.
  9. Click **Update**.
  10. Click **Save** directly to the master configuration and then restart server or cluster to pick up the changes.

  Attention: The CEA samples package includes a wsadmin (Jython) script library that you can use to simplify the development and testing of scripts that automate configuration changes. For further information, see the wsadmin (Jython) scripting procedures for CEA information.
- Use a custom application router.
  1. In the administrative console, click **Environment** > **SIP application routers**. The table displays a list of available application routers, including the DAR component.
  2. Select a custom SIP application router from the list, or click **New** to create a new one. The Configuration tab shows the name of the application router and the provider name of the application router. The provider name of the application router must be set to the custom application router implementation fully qualified class name.
  3. Place the Java archive (JAR) file in the server class path. For example, place the JAR file in the *java_home*/lib/ext directory, and ensure this directory is included in the class path for the server.

     gotcha: Do not add this path to the **Servers** > **Server Types** > **Websphere application servers** > ***server_name*** > **Sip Container** > **Java and Process Management** > **Process Definition** > **Java Virtual Machine** > **Classpath** entry, because this can cause conflicts.

4. Enter or edit the information in the required fields as needed. The required fields are identified with an asterisk (*).
5. Click **Apply** and then click **Save** directly to the master configuration.
6. To change the target of an application router, click on a router name, and check to see if a target is listed. If a target is listed, select a target and then click the **Move to Application Router** button. The drop-down menu lets you select another application router.
7. Click **Save** directly to the master configuration and then restart server or cluster to pick up the changes.

Attention: The CEA samples package includes a wsadmin (Jython) script library that you can use to simplify the development and testing of scripts that automate configuration changes. For further information, see the wsadmin (Jython) scripting procedures for CEA information.

- Use custom properties to configure the SIP application router. You can use the following custom properties to configure a DAR or a custom application router. These custom properties override the administrative console settings.
  1. In the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name***.
  2. Under **Container Settings**, expand **SIP Container Settings** and click **SIP container**.
  3. Under **Additional Properties**, click **Custom properties**, then click **New**.
  4. For the DAR, use the **javax.servlet.sip.ar.dar.configuration** custom property. This property specifies the location of the DAR properties file that defines the order in which the application router sends SIP requests to applications as described in Appendix C of the JSR 289 specification.
  5. For the custom application router, use the **javax.servlet.sip.ar.spi.SipApplicationRouterProvider** custom property. This property specifies the custom application router implementation fully qualified class name as described in section 15.4.2 of the JSR 289 specification. The custom application router implementation class defines the order in which the application router sends SIP requests to applications.
  6. Click **Save** directly to the master configuration and then restart server or cluster to pick up the changes.

Results

You have successfully configured a SIP application router.

SIP application router collection

Use this page to create and delete SIP application routers. The Session Initiation Protocol (SIP) application router allows you to select the order in which SIP applications are run at an initial SIP request.

To view this administrative console page, click **Environment > SIP application routers**. The table displays a list of available application routers, including the default application router (DAR).

New:

Click **New** to create a new SIP application router.

On the Configuation tab, specify the following required fields. Click **Apply** to save the entry or **Reset** to clear.

- **SIP application router name** – Specify a logical name for the application router.
- **SIP application router provider name** – Specifies the provider name of the application router. This is defined in the custom application router jar file, which must be in the application server's classpath.

Delete:

To delete a listed application router, select it, then click **Delete**. The DefaultSIPApplicationRouter cannot be deleted.

SIP Application Router Name:

Specifies the SIP application router name that you entered on the Configuration tab or the Default SIP Application Router.

SIP application router settings

Use this page to configure SIP application router container settings. The Session Initiation Protocol (SIP) application router allows you to select the order in which SIP applications are triggered. You can use the default application router (DAR) or specify a custom application router adhering to the SIP Servlet specification.

To view this administrative console page, click **Environment > SIP application routers**, then click *SIP_application_router_name* from the list displayed.

SIP application router name:

Enter a logical name representing the application router. If you selected the default application router, this field cannot be edited.

SIP application router provider name:

Specifies the provider name of the application router. This is defined in the custom application router jar file, which must be in the application server's classpath. This field is not visible when you edit the DAR.

Targets:

Specifies a list of server and cluster targets that are associated with an application router. By default, all servers and clusters defined in the system are listed as targets of the DAR. Each target can only be associated with one application router. For the DAR, you can also set the order in which the applications are run on each target.

Move to Application Router:

Specifies the **Move to Application Router** button, which allows you to change the targets of an application router. Each application router can have a set of targets to choose from. Click this button to move the selected target to another application router.

Target:

Specifies the target associated with an application router. Click the target name to view more information about the target server, such as whether to use a DAR configuration file or to specify the application order.

Full name:

Specifies the full name of the target. This can be helpful to distinguish each target. For example, if you have multiple targets named "server1" on different nodes, you can tell them apart by looking at the **node=** part of the Full Name.

Application routing order settings:

Use this page to specify whether the Default SIP Application Router (DAR) must rely on the basic application startup order or advanced application routing rules (DAR configuration).

To view this administrative console page, click **Environment > SIP application routers > DefaultSIPApplicationRouter > *target***.

Basic application startup order:

Specifies to use application startup order weights to determine the application routing order.

Click the link to go to the Startup order page where you can specify the application order weight for each application on the target server. Lower weighted values take precedence.

Advanced application routing rules (DAR configuration):

Specifies to use routing rules when determining the application routing order.

Click the link to go to the DAR configuration file rules page where you can edit, import, or view the rules to be used by the SIP application router for this target server.

Application startup order settings:

Use this page to define the order in which Session Initiation Protocol (SIP) requests are routed to applications.

To view this administrative console page, click **Environment > SIP application routers > DefaultSIPApplicationRouter > *target* > Configure application startup order**.

Startup order weight:

Specifies to manually order the installed applications using the Startup Weight column in the Application order table.

The Startup order weight determines the order in which the SIP application router sends SIP requests to applications. These values also determine the startup order of applications after a server restart. The lower values start first.

Application name:

Specifies the name of the SIP application whose startup order weight you can modify.

SIP container custom properties:

You can add any of the following custom properties to the configuration settings for a Session Initiation Protocol (SIP) container.

To specify custom properties for a specific SIP container, navigate to the custom properties page, and then specify a value for the custom property.

Important: The custom properties are supported as the primary method of configuration. Therefore, if a custom property is set and then you set the corresponding setting in the administrative console, the custom property value is used.

1. In the administrative console, expand **Servers > Server Types > WebSphere application servers >** *server_name* to open the configuration tab for the server.
2. From **Container settings**, expand **SIP Container settings**, and click **SIP container**.
3. From **Additional properties**, select **Custom Properties** > **New**.
4. On the settings page, type the custom property to configure in the **Name** field, and then type the value of the custom property in the **Value** field.

5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

The following list of SIP container custom properties is provided with the product. These properties are not shown on the settings page for the container.

You can define the following SIP container custom properties that are provided with the product. These properties are not shown on the settings page for the container.

- “auth.int.enable” on page 2442
- “com.ibm.sip.sm.lnm.size” on page 2442
- “com.ibm.webspehere.sip.security.digest.ldap.cachecleanperiod” on page 2442
- “com.ibm.websphere.sip.security.tai.usercachecleanperiod” on page 2442
- “com.ibm.ws.sip.key.set” on page 2442
- “com.ibm.ws.sip.tai.DisableSIPBasicAuth” on page 2442
- “DigestPasswordServerClass” on page 2443
- “enable.system.headers.modify” on page 2443
- “end.of.service.replication” on page 2443
- “immediate.replication” on page 2443
- “javax.servlet.sip.ar.dar.configuration” on page 2443
- “javax.servlet.sip.ar.spi.SipApplicationRouterProvider” on page 2443
- “javax.sip.bind.retries” on page 2444
- “javax.sip.bind.retry.delay” on page 2444
- “javax.sip.detect.pre.escaped.params” on page 2444
- “javax.sip.force.connection.reuse” on page 2444
- “javax.sip.hide.message.body” on page 2445
- “javax.sip.hide.message.headers” on page 2445
- “javax.sip.hide.request.uri” on page 2445
- “javax.sip.OUTBOUND_PROXY” on page 2445
- “javax.sip.PATH_MTU” on page 2446
- “javax.sip.stat.report.interval” on page 2446
- “javax.sip.trace.msg.in” on page 2446
- “javax.sip.trace.msg.out” on page 2446
- “javax.sip.transaction.invite.auto100” on page 2446
- “javax.sip.transaction.timer.a” on page 2447
- “javax.sip.transaction.timer.b” on page 2447
- “javax.sip.transaction.timer.cancel” on page 2447
- “javax.sip.transaction.timer.d” on page 2447
- “javax.sip.transaction.timer.e” on page 2448
- “javax.sip.transaction.timer.f” on page 2448
- “javax.sip.transaction.timer.g” on page 2448
- “javax.sip.transaction.timer.h” on page 2449
- “javax.sip.transaction.timer.i” on page 2449
- “javax.sip.transaction.timer.invite.server” on page 2449
- “javax.sip.transaction.timer.j” on page 2449
- “javax.sip.transaction.timer.k” on page 2450

- “javax.sip.transaction.timer.non.invite.server” on page 2450
- “javax.sip.transaction.timer.t1” on page 2450
- “javax.sip.transaction.timer.t2” on page 2450
- “javax.sip.transaction.timer.t4” on page 2451
- “on.outgoing.message.replication” on page 2451
- “pws_atr_name” on page 2451
- “replicate.with.confirmed.dialog.only” on page 2451
- “sip.container.heartbeat.enabled” on page 2451
- “sip.jsr289.parse.address” on page 2452
- “SIP_RFC3263_nameserver” on page 2452
- “thread.message.queue.max.size” on page 2453
- “weight.overload.watermark” on page 2453

auth.int.enable:

Specifies the **auth-int** quality of protection (QOP) for digest authentication. Digest authentication defines two types of QOP: **auth** and **auth-int**. By default, **auth** is used. When this custom property is set to True, the highest level of protection is used, which is the **auth-int** QOP.

Data type	String
Default	False

com.ibm.sip.sm.lnm.size:

Specifies the number of logical names in the application server. Each SIP object that can be replicated, such as a SIP session, is associated with a logical name. All objects with the same logical name are replicated to the same back-up container. The proxy can route messages to the correct container using the logical name found in the message. The value must be greater than 1.

Data type	String
Default	10

com.ibm.webspehere.sip.security.digest.ldap.cachecleanperiod:

Specifies the clean Lightweight Directory Access Protocol (LDAP) cache period in minutes.

Data type	String
Default	120

com.ibm.websphere.sip.security.tai.usercachecleanperiod:

Specifies the clean security subject cache period in minutes.

Data type	String
Default	15

com.ibm.ws.sip.key.set:

Specifies the key to use for SIP flow token security. When a value is specified for this property, SIP flow token security is automatically enabled.

Data type String
Default There is no default value

com.ibm.ws.sip.tai.DisableSIPBasicAuth:

Specifies whether to allow basic authentication for SIP.

Data type String
Default False

DigestPasswordServerClass:

Specifies types of user registries that are supported, except LDAP. To configure DigestTAI without the LDAP user registry, complete the following steps.

1. Create a class that implements this interface: com.ibm.ws.sip.security.digest.DigestPasswordServer
2. Add the following property to the SIP container custom property:
   Default LdapPasswordServer
3. Ensure that all users declared by the impl class are declared in the user registry configured for the product security.

Data type String
Default impl

enable.system.headers.modify:

Specifies whether the application has access to headers that are otherwise restricted.

Data type String
Default False

end.of.service.replication: Specifies whether changes are buffered until the thread for a siplet is about to end. If the value is set to `true`, then each change is buffered until the thread for the siplet is about to end.

Data type Boolean
Default true

immediate.replication: Specifies whether each change is immediately sent to the Data Replication Service. When this property is set to `true`, when replication is issued from a non-SIP container thread, the replication is immediately performed on the calling thread. When this property is set to `false`, the changes are buffered, and replication does not occur until all changes are made.

Setting this property to true might have a negative impact on performance.

Data type Boolean
Default false

javax.servlet.sip.ar.dar.configuration:

Specifies the location of the default application router (DAR) properties file. The properties file defines the order in which the application router sends SIP requests to applications as described in Appendix C of the JSR 289 specification.

Data type	String
Default	Null

javax.servlet.sip.ar.spi.SipApplicationRouterProvider:

Specifies the custom application router implementation fully qualified class name as described in section 15.4.2 of the JSR 289 specification. The custom application router implementation class defines the order in which the application router sends SIP requests to applications.

Data type	String
Default	Null

javax.sip.bind.retries:

Specifies the amount of time, in milliseconds, between attempts to start the SIP channel if the SIP port is busy with another process during server startup.

Data type	String
Default	60

javax.sip.bind.retry.delay:

Specifies the delay, in milliseconds, between attempts to start the SIP channel if the SIP port is busy with another process during server startup.

Data type	String
Default	5000

javax.sip.detect.pre.escaped.params:

Specifies whether to prevent the container from re-escaping Uniform Resource Identifier (URI) parameters that were pre-escaped by the application.

Enabling this property provides the application with more control over escaping URI parameters, when calling the **javax.servlet.sip.SipFactory.createURI()** and the **javax.servlet.sip.SipURI.setParameter()** parameters.

By default, the container only escapes characters that it must encode according to the RFC 3261 25.1 specification. In some cases, however, escaping additional characters might be required. Due to a limitation in the JSR 116 (5.2.1) specification, the application cannot perform its own escaping. Because of this limitation, attempts by the application to encode URI parameters causes the container to re-encode the percent sign. If the value of this property is set to true, the container cannot re-encode the percent sign.

Setting the value to true is not in compliance with the JSR 116 (5.2.1) specification, but provides the application with greater control over URI parameter escaping. APAR PK37192 describes the problem and the workaround.

Data type	String
Default	False

javax.sip.force.connection.reuse:

Specifies whether to force reuse of inbound connections for outbound requests. This custom property is only relevant for stream transports, such as Transmission Control Protocol (TCP) and Transport Layer Security (TLS). Disabling this property causes the container to create a separate connection for outbound requests, even if an existing connection is already established to the same peer address. The connection is automatically reused if the top Via header in the inbound request contains an alias parameter. (http://www.ietf.org/internet-drafts/draft-ietf-sip-connect-reuse-07.txt)

Data type	String
Default	False

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.hide.message.body:

Specifies to hide message content in logs. Set the value of this property to `true` to remove the message body text from SIP messages printed in the log files. This property only affects the representation of the messages in log files.

Data type	String
Default	False

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.hide.message.headers:

Specifies to hide the specified message header field names in log files. The value of this property is a comma-separated list of header field names that you want removed from SIP messages printed in the log files. This property only affects the representation of the messages in log files.

Data type	String
Default	None

javax.sip.hide.request.uri:

Specifies to hide request URIs in log files. Set the value of this property to `true` to remove request URIs from SIP messages printed in the log files. This property only affects the representation of the messages in log files.

Data type	Boolean
Default	False

javax.sip.OUTBOUND_PROXY:

Specifies the fixed address for routing all outbound SIP messages. The format is *address:port/transport*, such as `1.2.3.4:5065/tcp`.

Note: Do not use this property if the container is fronted by an application server SIP proxy.

Data type	String
Default	null

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.PATH_MTU:

Specifies the maximum transmission unit, in bytes, for outbound User Datagram Protocol (UDP) requests. The SIP stack measures the size of a request before sending it out on the UDP channel. If the request is larger than the value specified for **PATH_MTU-200** (1300 bytes by default), then the transport is switched from UDP to TCP before transmission. Increase this value to send larger requests over the UDP channel; however, messages might be truncated or dropped. See the RFC 3261-18.1.1 specification for details.

Data type	String
Default	1500

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.stat.report.interval:

Specifies the amount of time, in milliseconds, for reporting dispatch and timer statistics to a system.out file. A value of zero indicates no report.

Data type	String
Default	0

javax.sip.trace.msg.in:

Specifies whether to print incoming messages to a system.out file.

Data type	String
Default	False

javax.sip.trace.msg.out:

Specifies whether to print outbound messages to a system.out file.

Data type	String
Default	False

javax.sip.transaction.invite.auto100:

Specifies whether to automatically reply to invite requests with a `100 Trying` response. Disabling this property might increase the number of invite retransmissions.

Data type	String
Default	True

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.a:

Specifies, for UDP only, the amount of time, in milliseconds, before retransmitting invite requests for timer A for the RFC 3261 specification. This property is relevant for the invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.b:

Specifies the amount of time, in milliseconds, for the invite client transaction timeout timer (timer B) for the RFC 3261 specification.

Data type	String
Default	64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.cancel:

Specifies the amount of timer, in milliseconds, for the timer to keep the cancelled client transaction in the proceeding state before completing the cancelled transaction for the RFC 3261 9.1 specification. This property is relevant for the invite client transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

javax.sip.transaction.timer.d:

Specifies the wait time, in milliseconds, before retransmission of the invite response for timer D for the RFC 3261 specification. This property is relevant for the invite client transaction.

Data type	String
Default	32000

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.e:

Specifies, for UDP only, the amount of time, in milliseconds, before the retransmission of the initial non-invite request for timer E for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.f:

Specifies the amount of time, in milliseconds, for the non-invite transaction timeout timer (timer F) for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.g:

Specifies the amount of time, in milliseconds, before retransmission of an initial invite response for timer G for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.h:

Specifies the amount of time, in milliseconds, to wait for an acknowledgement (ACK) receipt for timer H for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.i:

Specifies the amount of time in milliseconds to wait for an ACK retransmission for timer I for the RFC 3261 specification. This property is relevant for the invite server transaction.

Data type	String
Default	javax.sip.transaction.timer.t4

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.invite.server:

Specifies the amount of time, in milliseconds, for the timer to keep the invite server transaction in the complete state. This timer is not defined in the RFC specification.

To avoid creating a new server transaction when a client retransmits an invite request, keep the completed server transaction for a period of time before removing invite retransmissions. This timer is started when the transaction changes to the terminated state. When the timer completes, the transaction is removed.

Data type	String
Default	32000

javax.sip.transaction.timer.j:

Specifies the amount of time in milliseconds to wait for non-invite request retransmission for timer J for the RFC 3261 specification. This property is relevant for the non-invite server transaction.

Data type String
Default 64*javax.sip.transaction.timer.t1

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.k:

Specifies the amount of time, in milliseconds, to wait for non-INVITE response retransmissions for timer K for the RFC 3261 specification. This property is relevant for the non-invite client transaction.

Data type String
Default javax.sip.transaction.timer.t4

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.non.invite.server:

Specifies the amount of time, in milliseconds, for an Application Programming Interface (API) timer for the application to respond to a non-invite request. This property is relevant for non-invite server transactions.

This timer is not defined in the RFC specification. This property is needed to stop the transaction if the application does not generate a final response to the request. The timer starts when the request arrives in the stack and stops when a response is generated by the application. If no response is generated before the timer stops, then the transaction completes.

Data type String
Default 34000

javax.sip.transaction.timer.t1:

Specifies the amount of time, in milliseconds, for a network round trip delay for timer T1 for the RFC 3261 specification. The value is used as a base for calculating some timers and is relevant for all types of transactions, such as client, server, invite, and non-invite transactions.

Data type String
Default 500

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.t2:

Specifies the maximum time in milliseconds before retransmitting non-invite requests and invite responses for timer T2 for the RFC 3261 specification.

Data type	String
Default	4000

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

javax.sip.transaction.timer.t4:

Specifies the maximum amount of time, in milliseconds, for a message to remain in the network. This value is used as a base for calculating other timers for timer T4 for the RFC 3261 specification.

Data type	String
Default	5000

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

on.outgoing.message.replication: Specifies whether changes are buffered until a siplet issues a request.send() or response.send() call. If the value is set to `true`, then each change is buffered until a siplet issues a request.send() or response.send() call.

Data type	Boolean
Default	false

pws_atr_name:

Specifies the LDAP attribute name that stores the user password.

Data type	String
Default	userpassword

replicate.with.confirmed.dialog.only:

Specifies whether to replicate the application session, even when no dialogs are confirmed. If the value is set to `false`, then the application session is replicated immediately after the session is created. Otherwise, the application session is only replicated when an associated dialog is confirmed.

Data type	String
Default	False

sip.container.heartbeat.enabled:

Specifies whether or not SIP network outage detection is enabled for the SIP container. SIP network outage detection allows the SIP proxy to send keepalive messages to the SIP container if the value of this property is set to `true`.

If the value is set to `false` for the SIP container, then this property has no effect on the SIP proxy. However, if the value is set to `true` for the SIP container, the value should also be set to `true` for the SIP proxy to ensure that keepalive messages are answered at the SIP container and not presented to the application.

Data type	String
Default	true

sip.jsr289.parse.address:

Specifies to use the SIP Servlet Specification 1.1, JSR 289 required format for createRequest() and createAddress() methods.

Note: The JSR 289 API requires that for any SIP URI that contains address parameters, you must enclose the SIP URI in angle brackets. The default behavior of the sip.jsr289.parse.address property is compliant with JSR 289 and correctly parses the address parameter as if it belongs to the SIP address. For example, when the property is set to `false`, the SIP address, `sip:fred@acme.com;param1=1`, is converted to `<sip:fred@acme.com;param1=1>`. When the property is set to `true`, the SIP address `sip:fred@acme.com;param1=1`, is converted to `<sip:fred@acme.com;>param1=1`.

Data type	String
Default	True

SIP_RFC3263_nameserver:

Specifies whether to allow a SIP URI to be resolved through Domain Name System (DNS) into the IP address, port, and transport protocol of the next hop.

The value of the property is a string containing one or two address and port tuples, where two tuples are separated by a space. The following examples specify a one address and port tuple or a two address and port tuple.

dottedDecimalAddress@.port

hostname.domain@port

IPV6address@port

The following example values represent a single tuple.

- 1.2.3.4@53
- example.com@53
- a:b:c::d@53

The following example values represent two tuples separated by a space.

- 1.2.3.4@53 example.com@53
- a:b:c::d@53 9.32.211.14@53

Data type	String
Default	null

depfeat: This custom property is deprecated. Do not use this custom property unless you are running in a mixed cell environment that includes at least one core group that contains a mixture of Version 7.0 and Version 6.x processes.

mixv: If you are running in a mixed cell environment, and you have core groups that contain a mixture of Version 7.0 and Version 6.x processes, you must continue to use this custom property.

thread.message.queue.max.size:

Specifies the maximum number of events allowed in the container threads queue. When this number is exceeded, the proxy server is notified that the container is overloaded and requests for new sessions are not accepted. Instead, the container returns an error message that indicates that the container is temporarily unavailable.

This value represents the total number of messages for all queues and reflects the state of the CPU. When the CPU approaches 100%, the maximum value for this custom property is reached quickly. Configure your system to limit the queue size and prevent the queue from reaching this threshold.

Data type	String
Default	1000

weight.overload.watermark:

Specifies the threshold value for the internal weight calculated by the container. When the container calculates the internal weight to be higher than the value specified, an overloaded container becomes available for service again.

This custom property represents a percentage of the maximum internal weight, such as 30 percent when the default value is set. When the high-water mark, or maximum threshold, is exceeded, the container waits until the weight drops below the maximum weight. This value cannot exceed `10`.

Data type	String
Default	3

Default application router rule collection:

Use this page to view or modify default application router (DAR) routing rules or import a new DAR configuration file. The DAR is a Session Initiation Protocol (SIP) application router that you can use to select the order in which SIP applications are triggered.

To view this administrative console page, click **Environment > SIP application routers > DefaultSIPApplicationRouter > *target* > Configure routing rules**.

The DAR configuration file contains the routing definitions for the application routing order of the DAR. You can upload a new DAR configuration file and view the definitions in the existing DAR configuration file. You can also use the New, Delete, Move up, and Move down buttons to edit rules individually. Use the **Method** menu list to filter the list of displayed rules by method name. Select **All Methods** to view all rules currently in the DAR configuration file.

Table 230. Button descriptions. This table shows button descriptions.

Button	Resulting action
New	Click to add a new DAR rule.

Table 230. Button descriptions (continued). This table shows button descriptions.

Button	Resulting action
Delete	Click to delete the selected rules.
Move Up / Move Down	Click to move the selected rule up or down in the list of rules. Rules must remain grouped by the method name; rules only move up or down within the method group. For example, if you try to move up an INVITE method, but the previous method is an ACK method, then this INVITE method is already at the top of the INVITE method group, and cannot move any higher.
Import	Click to open the file import window. Browse to the new DAR configuration file. **Attention:** Importing a new configuration file overwrites all existing rules.

Method:

Specifies the name of the SIP method used in the request.

Application:

Specifies the name of the application that the application router sends the SIP request to.

Subscriber identifier:

Specifies the identity of the subscriber that the DAR returns. The DAR can return any header in the SIP request. For example, if you specify `DAR:From` , the SIP URI is returned in the From header. The DAR can also return any string.

Routing region:

Specifies the routing region, which can be any of the following strings:
- ORIGINATING
- TERMINATING
- NEUTRAL

Route modifier:

Specifies the route modifier, which can be any of the following values:
- ROUTE
- ROUTE_BACK
- NO_ROUTE

Default application router rule settings

Use this page to edit the details of an application router rule.

To view this administrative console page, click **Environment > SIP application routers > DefaultSIPApplicationRouter > *target* > Configure routing rules**. Select a routing rule to edit or click **New**.

Routing rules are used to determine how the default SIP application router routes SIP requests to the installed applications.

Method:

Specifies the name of the SIP method in the request.

Select a method from the existing method list, or specify a new method name.

Application:

Specifies the name of the application that the application router sends the request to.

Subscriber identifier:

Specifies the identity of the subscriber that the DAR returns. The DAR can return any header in the SIP request. For example, if you specify `DAR:From` , the SIP URI is returned in the From header. The DAR can also return any string.

Select a subscriber identifier from the list of existing identifiers, or specify a new subscriber identifier name.

Routing region:

Specifies the routing region, which can be any of the following strings:

- ORIGINATING
- TERMINATING
- NEUTRAL

SIP URI:

Specifies the SIP URI that indicates the route as returned by the application router. This value can be an empty string.

Route modifier:

Specifies the route modifier, which can be any of the following values:

- ROUTE
- ROUTE_BACK
- NO_ROUTE

State info:

Specifies the stateInfo object, which the application router uses internally. For more information about this value, see the SIP servlet specification.

Configuring multihomed hosting

The SIP container can accept from the SIP proxy a list of outbound interfaces and expose it to any SIP application.

Before you begin

Multihomed hosting is configured at the WebSphere SIP proxy after the multihomed environment is set up. The multihome topology may include setting up multiple networks (routers, switches, etc.), multiple load balancers (if more than one proxy server needs to be configured for each virtual IP), and multiple network cards on each of the available proxy servers. After you install the network cards and configure the loopback addresses, set up a separate SIP proxy channel chain in each SIP proxy for each available network interface.

Attention: You can only configure the SIP proxy server to support multiple interfaces. The SIP container does not support this capability.

About this task

Multihoming allows you to have a single application communicate with different user agent clients (UAC) and user agent servers (UAS) on different networks.

The application queries the SIP container to determine the list of available outbound interfaces using standard procedures defined by JSR 289. This is done through a context attribute that is maintained in the container (through protocol exchanges with all the available SIP proxies). This attribute is javax.servlet.sip.SipServlet.OUTBOUND_INTERFACES, which is defined to be `javax.servlet.sip.outboundInterfaces`. This attribute contains all the available interfaces. The sample code in the Example section shows how to access the attribute from the application.

After the interfaces on each SIP proxy are configured, follow the steps in the procedure to control the routing of outbound messages. If more than one proxy is being used, it is important that each proxy be configured identically.

When an application does not specify an interface to use for sending outbound requests, the default interfaces are used by the proxy. It is recommended that you set the default interfaces for every protocol. See step 5 for more information.

The administrator can optionally set three SIP proxy custom properties that define the chain name that define the appropriate interface to use if the SIP application does not call the setOutboundInterface method. If these custom properties are not set and the setOutboundInterface method is not used, the interface that will be used for outbound requests cannot be definitively determined.

The following procedure applies to a topology that contains a single proxy setup for multihomed hosting with more than one network interface.

Procedure

1. In the administrative console, expand **Servers > Server Types** and click **WebSphere proxy servers > *proxy_name***.
2. Under **Proxy Settings**, expand **SIP Proxy Server Settings** and click **SIP proxy server transports**.
3. On the Transport Chain panel, delete existing transport chain or chains that contain proxy host names that use an asterisk (*).
4. Add new transport chain names and specify the IP address or host name associated with the interface that the chain is configured to use. Proxy multihomed configurations require you to configure a transport for each proxy interface. When using proxy servers with a load balancer, ensure a transport for TCP exists along with the desired transport type for SIP traffic. For example, when a proxy server uses two interfaces, then a minimum of six proxy transport chains are required. Each proxy interface will have a UDP transport chain (2) configured using the load balancer cluster alias IP address, TCP transport chain (2), and specific transport chain (2) of desired protocol (UDP, TLS, etc.) to run SIP traffic.
   a. On the Transport chain panel, click **New**. The Create New Transport Chain wizard initializes. During the transport chain creation process, add a unique Transport chain name and select the proxy protocol template (UDP, TCP, or Secure) from the Transport chain template menu.
   b. Click **Next**.
   c. Select the **Use existing port** or **Create a new port** option. For new ports, provide the port name, host name, and port number. For the Host value, specify the IP address or specific host name. Do not use an asterisk (*) for the Host value.
   d. Click **Next** for Step 2.
   e. Review the summary of actions and click **Finish** for Step 3.
   f. At the top of the panel, click **Save** to save the changes to the master configuration and resynchronize with the nodes, if applicable.

5. Specify the default chain name to use on the proxy server. From the SIP proxy settings panel, custom properties can be set up to specify the appropriate default interface for each protocol. These interfaces are used to send outbound requests when an application does not specify which interface to use.
   a. In the administrative console, expand **Servers > Server Types** and click **WebSphere proxy servers > *proxy_name***.
   b. Under **Proxy Settings** , expand **SIP Proxy Server Settings** and click **SIP proxy settings > Custom properties**.
   c. Enter the appropriate chain name previously configured in step 4 (not the interface or host name) to configure the transports section of the SIP proxy settings. There is one custom property for each transport type.

Custom Property Name	Description
defaultUDPChainName	The default UDP chain name to use when setOutboundInterface is not called.
defaultTCPChainName	The default TCP chain name to use when setOutboundInterface is not called.
defaultTLSChainName	The default TLS chain name to use when setOutboundInterface is not called.

6. Recycle the proxy server.

Results

You have successfully configured SIP multihomed hosting, which enables your applications to route outbound SIP requests through more than a single outbound interface.

Example

The following sample code demonstrates how to acquire the available outbound interfaces and set the appropriate outbound interface on the session object.

```
....
import javax.servlet.sip.SipServlet;
import javax.servlet.sip.SipSession;
....

protected void doInvite(SipServletRequest req1) throws ServletException, IOException
{
 ...
 // This block of code handles setting of the outbound interface.
         SipSession sipSession = req1.getSession();
 javax.servlet.ServletContext  context = getServletContext();
 java.util.List list = (java.util.List)context.getAttribute(javax.servlet.sip.SipServlet.
OUTBOUND_INTERFACES);
 SipURI uri = getProtocolInterface ("udp", list);

 if (uri != null)
 {
  InetSocketAddress inetSocketAddr = new InetSocketAddress(uri.getHost(), uri.getPort());
  sipSession .setOutboundInterface(inetSocketAddr);
 }
 ...
}

// This method simply pulls out the first interface in the list for the specified protocol
private SipURI getProtocolInterface(String transport, List outboundInterfaceList)
{
 SipURI uri = null;
 Iterator iterator = outboundInterfaceList.iterator();
```

```
 while (iterator.hasNext())
 {
  SipURI tempUri = (SipURI)iterator.next();

  if (tempUri.getTransportParam().equals(transport) == true)
  {
   uri = tempUri;
   break;
  }
 }

 return (uri);
}
```

Multihomed hosting:

SIP can support the ability to route outbound SIP requests through more than a single interface with the multihomed host feature of JSR 289.

In a multihomed host environment, the SIP container has the ability to select a particular outbound interface for routing messages. The SIP container can accept from the SIP proxy a list of outbound interfaces and expose it to any SIP application. This functionality is for applications that require tighter control over the outgoing request flow.

The following two methods can be used to select the outbound interface to use when sending requests, as defined in section 14.2 of the JSR 289 specification:

- `setOutboundInterface(java.net.InetAddress address)`
- `setOutboundInterface(java.net.InetSocketAddress address)`

A SIP application can obtain a list of available of SIP URIs it can send outbound requests on from the ServletContext attribute "javax.servlet.sip.outboundInterfaces," which is defined with the static string `javax.servlet.sip.SipServlet.OUTBOUND_INTERFACES`.

The application must set the interface on the Proxy, the ProxyBranch, or the SipSession objects before any outbound requests are sent. The interface is passed back in the attribute for outbound interfaces. The container then notifies the proxy which interface to send the outbound request on. Routing of non-request messages is controlled by other means, such as headers. For instance, a response message always flows through the same interface that the request arrived on.

Modifying the outbound routing of a request can affect all of the following SIP headers that are inserted into the outgoing SIP request, and subsequent responses in the dialog:

- Via header
- Contact headers
- Record-Route and Route headers
- Path header

Three SIP proxy custom properties specify the default chain names that define the appropriate interface to use for outbound requests. See the information on configuring multihomed hosting.

SIP with multihomed host is only supported in a distributed environment and must be configured at the WebSphere SIP proxy. A stand-alone SIP container does not support this capability.

Configuring multiple proxy servers using a load balancer in a multihomed environment

You can configure multiple proxy servers with multiple network interfaces using a load balancer in a multihomed environment.

Before you begin

Communications Enabled Applications (CEA) enables you to have multiple network interfaces on multiple proxy servers using a load balancer.

In a multihome environment, you need to define multiple virtual IP (VIP) alias addresses at the proxy server, and you need to define the proxy transport chains for each VIP alias address.

Note: Do not use an asterisk (*) for the transport chain names. Instead, use host names and IP addresses for the transport chains.

About this task

The following setup includes configuring a load balancer with two clusters, and two proxy servers with two VIP alias addresses defined. Ensure that the following steps are repeated for both proxy servers.

Procedure

1. In the administrative console, expand **Servers > Server Types** and click **WebSphere proxy servers > *proxy_name***.
2. Under **Proxy Settings**, expand **SIP Proxy Server Settings** and click **SIP proxy settings**.
3. Under Load balancer health checking, set the Load balancer members for IP address 1 and IP address 2. The IP address of the load balancer is used to source the SIP health checks.
4. Under Container facing network interface, specify the User Datagram Protocol (UDP) interface. The UDP interface specifies the network interface for all UDP data that goes to and from the backend SIP containers.
5. Click **Custom properties** and set the default protocol chain name types for each protocol type (UDP, TCP, and TLS). Specify the default chain name to use on the proxy server. From the SIP proxy settings panel, custom properties can be set up to specify the appropriate default interface for each protocol. These interfaces are used to send outbound requests when an application does not specify which interface to use.
   a. In the administrative console, expand **Servers > Server Types** and click **WebSphere proxy servers > *proxy_name***.
   b. Under **Proxy Settings** , expand **SIP Proxy Server Settings** and click **SIP proxy settings > Custom properties**.
   c. Enter the appropriate chain name to configure the transports section of the SIP proxy settings. There is one custom property for each transport type.

Custom Property Name	Description
defaultUDPChainName	The default UDP chain name to use when setOutboundInterface is not called.
defaultTCPChainName	The default TCP chain name to use when setOutboundInterface is not called.
defaultTLSChainName	The default TLS chain name to use when setOutboundInterface is not called.

6. In the administrative console, expand **Servers > Server Types** and click **WebSphere proxy servers > *proxy_name***.
7. Under **Proxy Settings**, expand **SIP Proxy Server Settings** and click **SIP proxy server transports**.
8. On the Transport Chain panel:
   a. Add two TCP transport chains using the host name or IP address for each network interface.
   b. Add two UDP transport chains using the host name or IP address for each network interface.

c. Add two UDP transport chains using the VIP alias addresses of the proxy servers so that the load balancer can communicate with the proxy servers. These two unique VIP alias addresses must be the same on both proxy servers.

9. Recycle the proxy server.

Results

You have successfully configured multihomed network interfaces for two proxy servers using a load balancer.

Chapter 26. Administering Startup beans

This page provides a starting point for finding information about startup beans.

Startup beans allow business logic to run when an application starts or stops.

Using startup beans

There are two types of startup beans: application startup beans and Module startup beans.

About this task

Note: The capabilities provided with startup singleton session beans (EJB 3.1 specification) causes the WebSphere Application Server proprietary startup beans function to be deprecated.

A module startup bean is a session bean that is loaded when an EJB Jar file starts. Module startup beans enable Java Platform Enterprise Edition (Java EE) applications to run business logic automatically, whenever an EJB module starts or stops normally. An application startup bean is a session bean that is loaded when an application starts. Application startup beans enable Java EE applications to run business logic automatically, whenever an application starts or stops normally.

Startup beans are especially useful when used with asynchronous bean features. For example, a startup bean might create an alarm object that uses the Java Message Service (JMS) to periodically publish heartbeat messages on a well-known topic. This enables clients or other server applications to determine whether the application is available. Refer to the Enabling an application to wait for a messaging engine to start article if you are using the default JMS provider.

Procedure

1. For Application startup beans, use the home interface, com.ibm.websphere.startupservice.AppStartUpHome, to designate a bean as an Application startup bean. For Module startup beans, use the home interface, com.ibm.websphere.startupservice.ModStartUpHome, to designate a bean as a Module startup bean.
2. For Application startup beans, use the remote interface, com.ibm.websphere.startupservice.AppStartUp, to define start() and stop() methods on the bean. For Module startup beans, use the remote interface, com.ibm.websphere.startupservice.ModStartUp, to define start() and stop() methods on the bean.

   The startup bean start() method is called when the module or application starts and contains business logic to be run at module or application start time.

   The start() method returns a boolean value. **True** indicates that the business logic within the start() method ran successfully. Conversely, **False** indicates that the business logic within the start() method failed to run completely. A return value of False also indicates to the Application server that application startup is aborted.

   The startup bean stop() methods are called when the module or application stops and contains business logic to be run at module or application stop time. Any exception thrown by a stop() method is logged only. No other action is taken.

   The start() and stop() methods must never use the TX_MANDATORY transaction attribute. A global transaction does not exist on the thread when the start() or stop() methods are invoked. Any other TX_* attribute can be used. If TX_MANDATORY is used, an exception is logged, and the application start is aborted.

   The start() and stop() methods on the remote interface use **Run-As** mode. **Run-As** mode specifies the credential information to be used by the security service to determine the permissions that a principal has on various resources. If security is on, the **Run-As** mode needs to be defined on all of the methods called. The identity of the bean without this setting is undefined.

There are no restrictions on what code the start() and stop() methods can run, since the full Application Server programming model is available to these methods.

3. Use an *optional* environment property integer, `wasStartupPriority`, to specify the start order of multiple startup beans in the same Java Archive (JAR) file. If the environment property is found and is the wrong type, application startup is aborted. If no priority value is specified, a default priority of `0` is used. It is recommended that you specify the priority property. Beans that have specified a priority are sorted using this property. Beans with numerically lower priorities are run first. Beans that have the same priority are run in an undefined order. All priorities must be positive integers. Beans are stopped in the opposite order to their start priority. The priority values for module startup beans and application startup beans are mutually exclusive. All modules will be started prior to the application being declared as "started" and therefore the start() methods for module startup beans within an application will be invoked prior to the start() methods for any application startup beans. Likewise, all application startup bean stop() methods for a specific Java Archive (JAR) file will be invoked prior to any module startup bean stop() methods for that JAR.
4. For module startup beans, the order in which EJB modules are started can be adjusted via the "Starting weight" value associated with each module
5. To control who can invoke startup bean methods via WebSphere Security do the following:
   a. Define the method permissions for the Start() and Stop() methods as you would for any EJB module. (See "Defining method permissions for EJB modules".)
   b. Ensure that the user that is mapped to the Security Role defined for the startup bean methods is the same user that is defined as the Server user ID within the User Registry.

What to do next

View the startup beans service settings.

Enabling startup beans in the administrative console

Enabling startup beans in the administrative console enables Java 2 Platform Enterprise Edition (J2EE) applications to run business logic automatically, whenever an application starts or stops normally.

About this task

Use the following steps to enable startup beans in the administrative console.

Procedure

1. Start the administrative console.
2. Select **Servers** > **Application Servers** > *server_name* > **Container Services** > **Startup beans service**.
3. Select the **Enable service at server startup** check box.
4. Click **Apply** to save the configuration.

What to do next

View the startup beans service settings.

Startup beans service settings

Use this page to enable startup beans that control whether application-defined startup beans function on this server. Startup beans are session beans that run business logic through the invocation of start and stop methods when applications start and stop. If the startup beans service is disabled, then the automatic invocation of the start and stop methods does not occur for deployed startup beans when the parent application starts or stops. This service is disabled by default. Enable this service only when you want to use startup beans. Startup beans are especially useful when used with asynchronous beans.

Note: The capabilities provided with startup singleton session beans (EJB 3.1 specification) causes the WebSphere Application Server proprietary startup beans function to be deprecated.

To view this administrative console page, click **Servers > Server types > WebSphere application servers >** *server_name*. Under **Container Settings**, expand **Container Services** then click **Startup beans service**.

Enable service at server startup

Specifies whether the server attempts to initiate the startup beans service.

Default
Cleared

Range

Selected
When the application server starts, it attempts to initiate the startup bean service automatically.

Cleared
The server does not try to initiate the startup beans service. All startup beans do not start or stop with the application. If you use startup beans on this server, then the system administrator must start the startup beans service manually or select this property, and then restart the server.

Chapter 27. Administering Transactions

This page provides a starting point for finding information about Java Transaction API (JTA) support. Applications running on the server can use transactions to coordinate multiple updates to resources as one unit of work, such that all or none of the updates are made permanent.

The product provides advanced transactional capabilities to help application developers avoid custom coding. It provides support for the many challenges related to integrating existing software assets with a Java EE environment. More introduction...

Administering the transaction service

You can view or change settings for the transaction service and manage active and prepared transactions. You can configure transaction properties to enable peer recovery of failed application servers in a cluster.

Procedure

- "Configuring transaction properties for an application server"
- "Managing active and prepared transactions" on page 2505
- "Managing transaction logging for optimum server availability" on page 2510
- "Displaying transaction recovery audit messages" on page 2514
- "Delaying the cancelling of transaction timeout alarms" on page 2515
- "Removing entries from the transaction partner log" on page 2515

Configuring transaction properties for an application server

You can view or change settings for the transaction service. For example, you can change the location or default file size of the transaction log files, change transaction timeout properties, or change heuristic-related properties.

About this task

The transaction service is a server runtime component that can coordinate updates to multiple resource managers to ensure atomic updates of data. Transactions are started and ended by applications or the container in which the applications are deployed.

You might undertake this task when you want to move the transaction logs to a different storage device, or when you have to change the transaction service settings. You must restart the application server to make configuration changes take effect.

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. The properties of the application server, *server_name*, are displayed in the content pane.
2. Click **[Container Settings] Container Services** > **Transaction Service**. The Transaction Service settings page is displayed.
3. Ensure that the Configuration tab is displayed.
4. Optional: To change the directory in which transaction logs are written, type the full path name of the directory in the **Transaction log directory** field. You can check the current runtime value of **Transaction log directory** by clicking the **Runtime** tab.

   If you do not enter a value for the **Transaction log directory**, the application server assumes a default location in the appropriate profile directory.

Note: If you change the transaction log directory, apply the change and restart the application server as soon as possible, to minimize the risk of problems occurring before the application server is restarted. For example, if there is a problem and the server fails with in-flight transactions, when the server restarts, it uses the new log directory and cannot automatically resolve in-flight transactions that were recorded in the old log directory.

You can also specify a size for the transaction logs, as described in the following step.

5. Optional: To change the size of transaction log files, modify the **Transaction log directory** field to include a file size setting. Use one of the following formats, where *directory_name* is the name of the transaction log directory and *file_size* is the disk space allocation for the transaction log files, specified in kilobytes (*n*K) or megabytes (*n*M). The minimum transaction log file size that you can specify is 64K. If you specify a value that is less than 64K, or you do not specify a value for the file size, the default value of 1M is used.

```
;file_size   <!-- This format keeps the default directory -->
directory_name;file_size
dir://directory_name/directory_name;file_size
/directory_name/directory_name;file_size
```

In a non-production environment, you can turn transaction logging off by entering `;0` in the **Transaction log directory** field (do not enter a directory name). Do not turn transaction logging off in a production environment because this prevents recovery after a system failure, and therefore data integrity cannot be guaranteed.

For more information about transaction log sizes, see “Managing transaction logging for optimum server availability” on page 2510.

6. Optional: Review or change the value of transaction timeout properties:

Total transaction lifetime timeout

The number of seconds to allow for a transaction that is started on this server, before the transaction service initiates timeout completion. If a transaction does not begin completion processing before this timeout occurs, it is rolled back. A value of `0` (zero) indicates that this timeout does not apply, and therefore the maximum transaction timeout is used instead. Application components can override the total transaction lifetime timeout for their transactions by setting their own timeout value.

If you are running your messaging system in non-ASF mode, you must make sure that this property is correctly configured with **NON.ASF.RECEIVE.TIMEOUT** so that unwanted transaction timeouts are avoided. See the related links for more details.

Maximum transaction timeout

The number of seconds a transaction that is propagated into this application server can remain inactive before it is ended by the transaction service. This value also applies to transactions that are started in this server, if their associated applications do not set a transaction timeout and the total transaction lifetime timeout is set to `0` (zero).

This value must be equal to, or greater than, the total transaction lifetime timeout. A value of `0` (zero) indicates that this timeout does not apply. In this situation, transactions that are affected by this timeout never time out.

Client inactivity timeout

The number of seconds after which a client is considered inactive and the transaction service ends any transactions associated with that client. A value of `0` (zero) indicates that there is no timeout limit.

7. Optional: Review or change heuristic-related properties:

Heuristic retry limit

The number of times that the application server retries a completion signal, such as commit or rollback. Retries occur after a transient exception from a resource manager or remote partner, or if the configured asynchronous response timeout expires before all Web Services Atomic Transaction (WS-AT) partners have responded.

Heuristic retry wait
The number of seconds that the application server waits before retrying a completion signal, such as commit or rollback, after a transient exception from a resource manager or remote partner.

Enable logging for heuristic reporting
Select this option to enable the application server to log “about to commit one-phase resource” events from transactions that involve a one-phase commit resource and two-phase commit resources.

Heuristic completion direction
Select the direction used to complete a transaction that has a heuristic outcome; either the application server commits or rolls back the transaction, or depends on manual completion by the administrator.

The heuristic completion direction property specifies how a transaction is completed in the following situations:

- The transaction manager reports a heuristic outcome for a last participant support (LPS) resource.
- The heuristic retry limit is exceeded during the recovery of a subordinate server in a distributed transaction.
- The transaction is imported from a Java EE Connector Architecture (JCA) provider.

This property applies only to transactions that are in the situations just described.

Accept heuristic hazard
Select this option to specify that all applications on this server accept the possibility of a heuristic hazard occurring in a two-phase transaction that contains a one-phase resource. This setting configures last participant support (LPS) for the server. If you do not select this option, you must configure applications individually to accept the heuristic hazard.

8. Optional: To change the default WS-Transaction specification level to use for outbound requests that include a web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) coordination context, select the specification level from the **Default WS-Transaction specification level** list.
9. Review or change other configuration properties, to suit your requirements. For more information about the properties of the transaction service, see the topic about Transaction service settings.
10. Click **OK**, then save your changes to the master configuration.
11. Stop, then restart, the application server.

What to do next

If you change the transaction log directory configuration property to an incorrect directory name, the application server restarts, but cannot open the transaction logs. Change the configuration property to a valid directory name, then restart the application server.

If you are running the application server as non-root, modify the permissions on the new transaction log location. To use peer recovery of transactions on a shared device with non-root users, make sure that your non-root users and groups have matching identification numbers across machines.

Transaction service settings

Use this page to specify settings for the transaction service. The transaction service is a server runtime component that can coordinate updates to multiple resource managers to ensure atomic updates of data. Transactions are started and ended by applications or the container in which the applications are deployed.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Transaction Service**.

Transaction log directory:

Specifies the name of a directory for this server where the transaction service stores log files for recovery. Optionally, you can specify the size of transaction log files.

Set this property to change the transaction log file directory for an application server only if the applications use distributed resources or XA transactions; for example, multiple databases and resources are accessed in a single transaction.

If you do not specify this directory during server configuration, the transaction service uses a default directory that is based on the installation directory: *app_server_root*/ tranlog/*cell_ name*/*node_ name*/*server_ name*.

When an application that runs on the application server accesses more than one resource, the application server stores transaction information in the product directory so that it can coordinate and manage the distributed transaction correctly. When there is a higher transaction load, storing persistent information in this way can slow the performance of the application server because it depends on the operating system and the underlying storage systems. To achieve better performance, designate a new directory for the log files on a separate, physically larger, storage system.

If your application server demonstrates one or more of the following symptoms, change the transaction log directory:
- CPU use remains low despite an increase in transactions
- Transactions fail with several timeouts
- Transaction rollbacks occur with the exception “Unable to enlist transaction”
- The application server stops in the middle of a run and must be restarted
- The disk that the application server is running on shows higher use

There are the following recommendations for a storage system for the log files:
- Store log files on a redundant array of independent disks (RAID)

  In RAID configurations, the task of writing data to the physical media is shared across the multiple drives. This technique yields more concurrent access to storage for persisting transaction information, and faster access to that data from the logs. Depending on the design of the application and storage subsystem, performance gains can range from 10% to 100%, or more in some cases.
- Do not store log files with the operation system I/O mode set to concurrent I/O (CIO)

  When you designate a transaction log directory, ensure that the file system uses only synchronous write-through and write serialization operations. Some operating systems, such as AIX JFS2, support an optional concurrent I/O (CIO) mode, where the file system does not enforce serialization of write operations. On these systems, do not use CIO mode for application server transaction recovery log files.

To specify the size of transaction log files, include a file size setting. Use one of the following formats, where *directory_name* is the name of the transaction log directory and *file_size* is the new disk space allocation for the transaction log files, specified in KB (*n*K) or MB (*n*M). The minimum transaction log file size that you can specify is 64K. If you specify a value that is less than 64K, or you do not specify a value for the file size, the default value of 1M is used.

```
;file_size   <!-- This format keeps the default directory -->
directory_name;file_size
dir://directory_name/directory_name;file_size
/directory_name/directory_name;file_size
```

For more information about transaction log sizes, see “Managing transaction logging for optimum server availability” on page 2510.

Data type String

Default	Directory name: *app_server_root*/tranlog/*cell_name*/*node_name*/*server_name* File size: 1MB
Recommended	Create a file system with at least three to four disk drives raided together in a RAID-0 configuration. Then, create the transaction log on this file system with the default size. When the server is running under load, check the disk input and output. If disk input and output time is more then 5%, consider adding more physical disks to lower the value.

Total transaction lifetime timeout:

The default maximum time, in seconds, allowed for a transaction that is started on this server before the transaction service initiates timeout completion. Any transaction that does not begin completion processing before this timeout occurs is rolled back.

This timeout is used only if the application component does not set its own transaction timeout.

The upper limit of this timeout is constrained by the maximum transaction timeout. For example, if you set a value of `500` for the total transaction lifetime timeout, and a value of `300` for the maximum transaction timeout, transactions will time out after 300 seconds.

If you set this timeout to `0`, the timeout does not apply and the value of the maximum transaction timeout is used instead.

Data type	Integer
Units	Seconds
Default	120
Range	0 to 2 147 483 647

Asynchronous response timeout:

Specifies the amount of time, in seconds, that the server waits for an inbound Web Services Atomic Transaction (WS-AT) protocol response before resending the previous WS-AT protocol message.

Data type	Integer
Units	Seconds
Default	30
Range	0 to 2 147 483 647

Client inactivity timeout:

Specifies the maximum duration, in seconds, between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back in this application server.

If you set this value to `0`, there is no timeout limit.

Data type	Integer
Units	Seconds
Default	60
Range	0 to 2 147 483 647

Maximum transaction timeout:

Specifies, in seconds, the upper limit of the transaction timeout for transactions that run in this server. This value should be greater than or equal to the value specified for the total transaction timeout.

This timeout constrains the upper limit of all other transaction timeout periods.

Table 231. Transaction timeout settings. The table shows how the different timeout settings apply to transactions running in the server.

Timeout setting	Transactions affected
Maximum transaction timeout	All transactions running in this server that are not affected by the total transaction lifetime timeout or an application component timeout. These transactions include transactions imported from outside this server, such as those imported from a client.
Total transaction lifetime timeout	All transactions that originated in this server that are not affected by an application component timeout, in other words, the associated application component does not set its own timeout.
Application component timeout	Transactions that are specific to an application component. You cannot set this transaction timeout using the administrative console. If the component is a container-managed bean, set this timeout in the deployment descriptor for the component. For example, you can use an assembly tool, such as the Rational Application Developer. If the component is a bean-managed bean, set this timeout programmatically by using the UserTransaction.setTransactionTimeout method.

If you set a timeout to `0`, that timeout does not apply, and is effectively disabled. If you set all timeouts to `0`, transactions never time out.

For example, consider the following timeout values:

Table 232. Example timeout values. The table lists different timeout settings and their values.

Timeout setting	Value
Maximum transaction timeout	360
Total transaction lifetime timeout	240
Application component timeout	60

In this example, transactions that are specific to the application component time out after 60 seconds. Other local transactions time out after 240 seconds, and any transactions that are imported from outside this server time out after 360 seconds. If you then change the application component timeout to `500`, application component transactions time out after 360 seconds, the value of the maximum transaction timeout. If you set the maximum transaction timeout to `0`, application component transactions time out after 500 seconds. If you remove the application component timeout, application component transactions time out after 240 seconds.

To determine the occurrence of a timeout quickly, and to prevent further resource locking, the application server prevents further transactional work on the transactional path where the timeout condition has taken place. This applies equally to attempting to undertake work under the current transaction context and to attempting to perform work under a different transactional context.

Data type Integer
Units Seconds
Default 300
Range 0 to 2 147 483 647

Heuristic retry limit:

Specifies the number of times that the application server retries a completion signal, such as commit or rollback. Retries occur after a transient exception from a resource manager or remote partner, or if the configured asynchronous response timeout expires before all Web Services Atomic Transaction (WS-AT) partners have responded.

If the application server abandons the retries, the resource manager or remote partner is responsible for ensuring that the resource or partner branch of the transaction is completed appropriately. The application server raises (on behalf of the resource or partner) an exception that indicates a heuristic hazard. If a commit request was made, the transaction originator receives an exception on the commit operation; if the transaction is container-initiated, the container returns a remote exception or Enterprise JavaBeans (EJB) exception to the EJB client.

During recovery of a subordinate server in a distributed transaction, when the number of heuristic retries is exceeded, the heuristic completion direction property specifies how the transaction is completed.

Data type	Integer
Default	0
Range	0 to 2 147 483 647 A value of 0 (the default) means try again indefinitely.

Heuristic retry wait:

Specifies the number of seconds that the application server waits before retrying a completion signal, such as commit or rollback, after a transient exception from a resource manager or remote partner.

Data type	Integer
Default	0
Range	0 to 2 147 483 647 A value of 0 means that the application server determines the retry wait; the server doubles the retry wait after every 10 failed retries.

Enable logging for heuristic reporting:

Specifies whether the application server logs about-to-commit-one-phase-resource events from transactions that involve both a one-phase commit resource and two-phase commit resources.

This property enables logging for heuristic reporting. If applications are configured to allow one-phase commit resources to participate in two-phase commit transactions, reporting of heuristic outcomes that occur at application server failure requires extra information to be written to the transaction log. If enabled, one additional log write is performed for any transaction that involves both one-phase and two-phase commit resources. No additional records are written for transactions that do not involve a one-phase commit resource.

Data type	Check box
Default	Cleared

Range

Cleared
The application server does not log "about to commit one-phase resource" events from transactions that involve a one-phase commit resource and two-phase commit resources.

Selected
The application server does log "about to commit one-phase resource" events from transactions that involve a one-phase commit resource and two-phase commit resources.

Heuristic completion direction:

Specifies the direction that is used to complete a transaction that has a heuristic outcome; either the application server commits or rolls back the transaction, or depends on manual completion by the administrator.

The heuristic completion direction property specifies how a transaction is completed in the following situations:

- The transaction manager reports a heuristic outcome for a last participant support (LPS) resource.
- The heuristic retry limit is exceeded during the recovery of a subordinate server in a distributed transaction.
- The transaction is imported from a Java EE Connector Architecture (JCA) provider.

This property applies only to transactions that are in the situations just described.

Data type Drop-down list
Default ROLLBACK
Range

COMMIT
The application server heuristically commits the transaction.

ROLLBACK
The application server heuristically rolls back the transaction.

MANUAL
The application server depends on an administrator to manually complete or roll back transactions with heuristic outcomes.

Accept heuristic hazard:

Specifies whether all applications on this server accept the possibility of a heuristic hazard occurring in a two-phase transaction that contains a one-phase resource. This setting configures last participant support (LPS) for the server. Last participant support is an extension to the transaction service that enables a single one-phase resource to participate in a two-phase transaction with one or more two-phase resources.

If the Accept heuristic hazard option is not selected, you must configure applications individually to accept the heuristic hazard. You can configure applications either when they are assembled, or following deployment by using the Last participant support extension pane.

Data type Check box
Default Cleared

Range

Selected
All applications deployed on the server accept the increased risk of an heuristic outcome.

Cleared
Applications must be individually configured to accept the increased risk of an heuristic outcome.

Enable file locking:

Specifies whether the use of file locks is enabled when opening the transaction service recovery log.

If you enable this setting, a file lock will be obtained before accessing the transaction service recovery log files. File locking is used to ensure that, in a highly available WebSphere Application Server deployment, only one application server can access a particular transaction service recovery log at any one time. This setting has no effect in a standard deployment where you have not configured high availability support.

Attention: This setting requires a compatible network file system, such as Network File System (NFS) version 4, to operate correctly.

Data type Check box
Default Selected

Enable transaction coordination authorization:

Specifies whether the secure exchange of transaction service protocol messages is enabled.

This setting has no effect unless you enable WebSphere Application Server security on the server.

Data type Check box
Default Selected

Default WS-Transaction specification level:

Specifies the default WS-Transaction specification level to use for outbound requests that include a Web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) coordination context.

You can choose from WS-Transaction 1.0, WS-Transaction 1.1 or WS-Transaction 1.2. For details of these specifications, see the topics about WS-AT support or WS-BA support in the application server.

The default WS-Transaction specification level is used if a level cannot be determined from the provider policy (the WS-Transaction WS-Policy assertion). This could be, for example, if the policy assertion is not available either from the WSDL of the target web service or from the WS-Transaction policy type of the client, or if the policy assertion is available but more than one specification level is applicable.

Data type Drop-down list
Default 1.0

External WS-Transaction HTTP(S) URL prefix:

Select or specify the external WS-Transaction HTTP(S) URL prefix.

Select or specify one of these fields if you are using an intermediary node, such as an HTTP server or Proxy Server for WebSphere, to send requests that comply with the Web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) protocols.

If WebSphere Application Server security is enabled and transaction coordination authorization is enabled, the HTTPS prefix is used. Otherwise, the HTTP prefix is used.

If the intermediary node is not a Proxy Server, the prefix must be unique for each server.

Select prefix:

Select this option to select the external endpoint URL information to use for WS-AT and WS-BA service endpoints from the list.

Data type Drop-down list
Default None

Specify custom prefix:

Select this option to specify the external endpoint URL information to use for WS-AT and WS-BA service endpoints in the field.

Use one of the following formats for the prefix, where *host_name* and *port* represent the intermediary node that is an HTTP or HTTPS proxy for the server.

`http://host_name:port`

`https://host_name:port`

Data type String
Default None

Manual transactions:

Specifies the number of transactions that await manual completion by an administrator.

If there are transactions awaiting manual completion, you can click the **Review** link to display a list of those transactions on the Transactions needing manual completion panel.

Data type Integer
Default 0

Retry transactions:

Specifies the number of transactions with some resources being retried.

If there are transactions with resources being retried, you can click the **Review** link to display a list of those transactions on the Transactions retrying resources panel.

Data type Integer
Default 0

Heuristic transactions:

Specifies the number of transactions that have completed heuristically.

If there are transactions that have completed heuristically, you can click the **Review** link to display a list of those transactions on the Transactions with heuristic outcome panel.

Data type	Integer
Default	0

Imported prepared transactions:

Specifies the number of transactions that are imported and prepared but not yet committed.

If there are transactions that have been imported and prepared but not yet committed, you can click the **Review** link to display a list of those transactions on the Transactions imported and prepared panel.

Data type	Integer
Default	0

Additional Properties:

Under Additional Properties you can click the **Custom properties** link to display or change custom properties for your WebSphere Application Server transaction service.

You use **Custom properties** to specify whether or not information messages are displayed on the administrative console and written to the `SystemOut.log` file upon transaction service recovery.

To find out more about WebSphere Application Server transaction service custom properties, see the related link.

Transactions needing manual completion:

Use this page to review transactions that need manual completion.

It is unusual for transactions to require manual completion. A transaction needs manual completion in the following circumstances:

1. An application was exploiting the last participant support to coordinate a single one-phase capable resource and one or more two-phase capable resources.
2. A failure occurred during the commit of the one-phase capable resource.
3. The transaction service heuristic completion direction is set to **Manual**, in the transaction service settings.

An administrator reviewing transactions in this state can review the outcome of any one-phase resources, by using facilities provided by the specific resource manager, then use this page to complete the transaction accordingly.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > [Content pane] *server_name* > **[Container Settings] Container Services** > **Transaction Service** > **Runtime** > **Manual transactions - Review**.

To list the resources used by a transaction, click the transaction local ID in the list displayed.

To act on one or more of the transactions listed, select the check boxes next to the transactions that you want to act on, then use the buttons provided.

Local ID

The local identifier of the transaction.

Global ID
The global identifier of the transaction.

Buttons

Commit
Heuristically commit the selected transactions.

Rollback
Heuristically roll back the selected transactions.

Transactions retrying resources:

Use this page to review transactions with resources being retried.

If the transaction manager has prepared resources, but has lost contact with the resource managers before committing them or aborting them, then the transaction manager retries the commit or rollback requests to the affected resource managers. The number of times and frequency that the transaction manager retries such commit or rollback requests is configured on the **Heuristic retry limit** and **Heuristic retry wait** properties of the transaction service settings.

An administrator can use this page to make the transaction service abandon the retries of one or more transactions. If a resource manager cannot be contacted, WebSphere Application Server relegates that resource manager to an in-doubt (prepared) state. The administrator then needs to use mechanisms specific to the resource manager to resolve the in-doubt status.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers >** [Content pane] *server_name* **> [Container Settings] Container Services** > **Transaction Service** > **Runtime** > **Retry transactions - Review**.

To list the resources used by a transaction, click the transaction local ID in the list displayed.

To act on one or more of the transactions listed, select the check boxes next to the transactions that you want to act on, then use the buttons provided.

Local ID
The local identifier of the transaction.

Status
The status of the transaction, shown as an integer value. The values correspond to the following status:

0 - active
1 - marked for rollback
2 - prepared
3 - committed
4 - rolled back
5 - unknown
6 - none
7 - preparing
8 - committing
9 - rolling back

Global ID
The global identifier of the transaction.

Buttons

Finish Stop retrying resources for the selected transactions.

Transactions with heuristic outcome:

Use this page to review transactions that completed with a heuristic outcome.

The page is provided for information purposes only. After you have reviewed the information in this page, then the only action required is to remove the transactions from the list. If you do not remove a transaction from the list, it is kept in the list for three days or until the server is shut down, whichever occurs first.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** [Content pane] *server_name* [Container Settings] **Container Services** > **Transaction Service** > **Runtime** > **Heuristic transactions - Review**.

To list the resources used by a transaction, click the transaction local ID in the list displayed.

To act on one or more of the transactions listed, select the check boxes next to the transactions that you want to act on, then use the buttons provided.

Local ID
The local identifier of the transaction.

Heuristic outcome
The outcome of the transaction.

Global ID
The global identifier of the transaction.

Buttons

Clear Remove the selected transactions from the list.

Transactions imported and prepared:

Use this page to review transactions that have been imported and prepared but not yet committed.

Under normal circumstances no administrative action is required for any of the transactions listed on this page. This page lists those transactions that are in a prepared state, but are being directed by an external transaction manager (for example, another WebSphere application server) from a transaction context that has been propagated.

Under aberrant circumstances, however, an administrator can configure WebSphere Application Server to resolve the transactions listed on this page independent of the external transaction manager. (This step might be necessary if, for example, the external transaction manager has become unavailable for an unacceptable period of time.)

Note: If the completion direction (commit or rollback) chosen administratively differs from the eventual direction of the external transaction manager, then the overall outcome of the transaction is not atomic and data corruption can result.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** [Content pane] *server_name* > **[Container Settings] Container Services** > **Transaction Service** > **Runtime** > **Imported prepared transactions - Review**.

To list the resources used by a transaction, click the transaction local ID in the list displayed.

To act on one or more of the transactions listed, select the check boxes next to the transactions that you want to act on, then use the buttons provided.

Local ID
The local identifier of the transaction.

Global ID
The global identifier of the transaction.

Buttons

Commit
Heuristically commit the selected transactions.

Rollback
Heuristically roll back the selected transactions.

Transaction resources:

Use this page to review resources used by a transaction.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** [Content pane] *server_name* > **[Container Settings] Container Services** > **Transaction Service** > **Runtime** > *transaction_type local_ID*.

Where:

- *transaction_type* is one of:
  - **Manual transactions - Review**
  - **Retry transactions - Review**
  - **Heuristic transactions - Review**
  - **Imported prepared transactions - Review**
- *local_ID* is the local ID of the transaction (as an active link in the list of transactions).

The details displayed depend on the resource provider.

Transaction service custom properties

WebSphere Application Server allows you to configure a number of custom properties for transaction services.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Transaction service custom properties can be specified in the administrative console. To use any of these custom properties click on **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Transaction Service** > **[Additional Properties] Custom Properties**.

You can define the following transaction service custom properties:

- DELAY_CANCELLING_ALARMS
- DISABLE_RECOVERY_AUDIT_LOGGING
- REMOVE_PARTNER_LOG_ENTRY

DELAY_CANCELLING_ALARMS: If the before completion stage of a transaction process is likely to include processes that could either take a long time to complete or could fail, then you might want the transaction to time out.

By default, transaction timeout alarms are cancelled prior to the before completion phase of the transaction begins. The DELAY_CANCELLING_ALARMS custom property allows the before completion phase of the transaction to be encompassed within the transaction timeout period. To do this, set the custom property on the application server.

Table 233. DELAY_CANCELLING_ALARMS custom properties. The table includes the data type, acceptable values, and default for the property.

Data type	Boolean
Acceptable values	TRUE, FALSE
Default	FALSE

DISABLE_RECOVERY_AUDIT_LOGGING: You can control whether information messages are displayed on the administrative console and written to the `SystemOut.log` file upon transaction service recovery. To do this, set the DISABLE_RECOVERY_AUDIT_LOGGING custom property for the transaction service for the server.

On distributed platforms the default is for information messages to appear both on the administrative console and in the `SystemOut.log` file during the recovery of transaction services. If you do not want these messages to be displayed you can use the DISABLE_RECOVERY_AUDIT_LOGGING custom property.

Table 234. DISABLE_RECOVERY_AUDIT_LOGGING custom properties. The table includes the data type, acceptable values, and default for the property.

Data type	Boolean
Acceptable values	TRUE, FALSE
Default	FALSE

REMOVE_PARTNER_LOG_ENTRY: You can remove entries from the transaction partner log file. To do this, set the REMOVE_PARTNER_LOG_ENTRY custom property for the transaction service on the server that owns the partner log.

As part of the transaction recovery process, the partner log is checked to establish which resources are needed. If you want to remove certain entries from the partner log, such as a resource that no longer exists, set this custom property on the application server that owns the transaction partner log containing the entries you want to remove.

The REMOVE_PARTNER_LOG_ENTRY custom property is effective only when both of the following situations apply.

- The application server is started in recovery mode.
- The application server has no transactions that currently require recovery. You can establish this by checking the `SystemOut.log` file.

Table 235. REMOVE_PARTNER_LOG_ENTRY custom properties. The table includes the data type, acceptable values, and default for the property.

Data type	Integer
Acceptable values	(one or more comma-delimited integer recovery ID)
Default	(null)

Managing active and prepared transactions

Use this task to manage active and prepared transactions that might need administrator action.

About this task

Under normal circumstances, transactions should run and complete (commit or roll back) automatically, without the need for intervention. However, in some circumstances, you might have to resolve a

transaction manually. For example, you might want to roll back a transaction that has become stuck polling a resource manager that you know will not become available again within the required timeframe.

Note: If you choose to complete a transaction on an application server, it is recorded as having completed in the transaction service logs for that server, so will not be eligible for recovery during server start up. If you complete a transaction, you are responsible for cleaning up any in-doubt transactions on the resource managers affected.

You can use the administrative console to display a snapshot of all the transactions in an application server that are in the following states:

Manual transactions
Transactions awaiting administrative completion. For each transaction, the local id or global id is displayed. You can choose to display information on each resource (specifically, which resource manager it is associated with) associated with the transaction. You can also choose to commit or roll back transactions in this state.

Retry transactions
Transactions with some resources being retried. For each transaction, the local id or global id is displayed, and whether the transaction is committing or rolling back. You can choose to display information on each resource (specifically, which resource manager it is associated with) associated with the transaction. You can also choose to finish (stop retrying) transactions in this state.

Heuristic transactions
Transactions that have completed heuristically. For each transaction, the local id or global id and the heuristic outcome is displayed. You can choose to display information on each resource (specifically, which resource manager it is associated with) associated with the transaction. You can also choose to clear the transaction from the list.

Imported prepared transactions
Transactions that have been imported and prepared but not yet committed. For each transaction, the local id or global id is displayed. You can choose to display information on each resource (specifically, which resource manager it is associated with) associated with the transaction. You can also choose to commit or roll back transactions in this state.

To manage the active and prepared transactions for an application server, use the administrative console to complete the following steps:

Procedure

1. Display the Transaction Service runtime page for application server:
   a. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers**.
   b. In the content pane, click the name of the application server.
   c. In the content pane, click the **Runtime** tab.
   d. Under Additional Properties, click **Transaction Service**.

   The page displays values for the runtime properties of the transaction service, including the number of transactions in the active and prepared states.
2. To display a snapshot of the transactions in a specific state, click **Review** in the field label.
3. Optional: To display information about the resources associated with a transaction, click the name of the transaction.
4. Optional: To act on a transaction, select the check box provided on the entry for the transaction, then click one of the buttons provided. Alternatively, to act on all transactions, select the check box in the header of the transactions table, then click a button.

Managing active and prepared transactions by using wsadmin scripting

You can use wsadmin scripting to manage active and prepared transactions that might need administrator action.

Before you begin

Before you start this task, the wsadmin scripting client must be running.

About this task

In normal circumstances, transactions should run and complete (commit or roll back) automatically, without the need for intervention. However, in some circumstances, you might have to resolve a transaction manually. For example, you might want to roll back a transaction that is stuck polling a resource manager that you know will not become available again in the required time frame.

Note: If you choose to complete a transaction on an application server, it is recorded as having completed in the transaction service logs for that server, so it is not eligible for recovery during server start up. If you complete a transaction, you are responsible for cleaning up any in-doubt transactions on the resource managers affected.

For more information about the TransactionService and Transaction MBeans, see the application programming interface (API) documentation.

Procedure

- You can use the TransactionService Managed Bean (MBean) to list transactions in various states by invoking one of the following methods:

  listOfTransactions
  Lists all non-completed transactions. Never attempt to alter the state of active transactions (for example, by using commit or rollback).

  listManualTransactions
  Lists transactions awaiting administrative completion. You can commit or roll back transactions in this state.

  listRetryTransactions
  Lists transactions with some resources being retried. You can finish (stop retrying) transactions in this state.

  listHeuristicTransactions
  Lists transactions that have completed heuristically. You can clear these transactions from the list.

  listImportedPreparedTransactions
  Lists transactions that have been imported and prepared but not yet committed. You can commit or roll back transactions in this state.

  Each entry in the returned list contains the following attributes:
  - **Local Transaction Identifier**
  - **Status**, which can be interpreted by calling getPrintableStatus on the Transaction MBean.
  - **Global Transaction Identifier**
  - **Heuristic Outcome**, which can take one of the following values:
    - 8 (HEURISTIC_COMMIT)
    - 9 (HEURISTIC_ROLLBACK)
    - 10 (HEURISTIC_MIXED)
    - 11 (HEURISTIC_HAZARD)

- You can use the TransactionService MBean to gather more information about the properties of the transaction service, by obtaining the following attributes:

  transactionLogDirectory
  The directory for this server where the transaction service stores log files for recovery.

  totalTranLifetimeTimeout
  The default maximum time, in seconds, allowed for a transaction that is started on this server before the transaction service initiates timeout completion. Any transaction that does not begin completion processing before this timeout occurs is rolled back. This value applies only to container-managed transaction (CMT) beans.

  asyncResponseTimeout
  The time, in seconds, that the server waits for an inbound Web Services Atomic Transaction (WS-AT) protocol response before resending the previous WS-AT protocol message.

  enableFileLocking
  Specifies whether the use of file locks is enabled when opening the transaction service recovery log.

  enableProtocolSecurity
  Specifies whether the secure exchange of transaction service protocol messages is enabled.

  clientInactivityTimeout
  The maximum duration, in seconds, between transactional requests from a remote client. Any period of client inactivity that exceeds this timeout results in the transaction being rolled back in this application server.

  heuristicRetryLimit
  The number of times that the application server retries a completion signal, such as commit or rollback. Retries occur after a transient exception from a resource manager or remote partner, or if the configured asynchronous response timeout expires before all Web Services Atomic Transaction (WS-AT) partners have responded.

  heuristicRetryWait
  The number of seconds that the application server waits before retrying a completion signal, such as commit or rollback, after a transient exception from a resource manager or remote partner.

  propogatedOrBMTTranLifetimeTimeout
  The upper limit of the transaction timeout, in seconds, for transactions that run in this server. This value must be greater than or equal to the total transaction timeout.

  LPSHeuristicCompletion
  The action to use to complete a transaction that has an heuristic outcome. Either the application server commits or rolls back the transaction, or the administrator must complete the transaction manually.

- You can use the Transaction MBean to commit, roll back, or finish a transaction, or remove a transaction from the list of heuristically completed transactions, depending on the state of the transaction, by invoking one of the following methods:

  commit
  Heuristically commits the transaction.

  rollback
  Heuristically rolls back the transaction.

  finish Stops retrying resources for the transaction.

  removeHeuristic
  Clears the transaction from the list.

- You can use the Transaction MBean to gather more information about a transaction, by invoking the following methods:

  getPrintableStatus
  Return the transaction status.

  getGlobalTranName
  Get the global identifier for the transaction.

  listResources
  List the resources for the transaction.

Example

The following script is an example of how to use the TransactionService and Transaction MBeans to work with manual transactions. Run the script only against an application server, and not against the deployment manager or node agent.

Example Jacl script:

```
# get the TransactionService MBean
set servicembean [$AdminControl queryNames type=TransactionService,*]

# get the Transaction MBean
set mbean [$AdminControl queryNames type=Transaction,*]

set input 0
while {$input >= 0} {
        # invoke the listManualTransactions method
        set tranManualList [$AdminControl invoke $servicembean listManualTransactions]

        if {[llength $tranManualList] > 0} {
                puts "----Manual Transaction details---------------"
                set index 0
                foreach tran $tranManualList {
                        puts "   Index= $index tran= $tran"
                        incr index
                }
                puts "----End of Manual Transactions ---------------"
                puts "Select index of transaction to commit/rollback:"
                set input [gets stdin]
                if {$input < 0} {
                        puts "No index selected, exiting."
                } else {
                        set tran [lindex $tranManualList $input]
                        set commaPos [expr [string first "," $tran ]-1]
                        set localTID [string range $tran 0 $commaPos]
                        puts "Enter c to commit or r to rollback Transaction $localTID"
                        set input [gets stdin]
                        if {$input=="c"} {
                                puts "Committing transaction=$localTID"
                                $AdminControl invoke $mbean commit $localTID
                        }
                        if {$input=="r"} {
                                puts "Rolling back transaction=$localTID"
                                $AdminControl invoke $mbean rollback $localTID
                        }
                }
        } else {
                puts "No Manual transactions found, exiting"
                set input -1
        }
        puts " "

}
```

Example Jython script:

```
import sys
def wsadminToList(inStr):
        outList=[]
        if (len(inStr)>0 and inStr[0]=='[' and inStr[-1]==']'):
                tmpList = inStr[1:-1].split(" ")
```

```
        else:
                tmpList = inStr.split("\n")  #splits for Windows or Linux
        for item in tmpList:
                item = item.rstrip();        #removes any Windows "\r"
                if (len(item)>0):
                        outList.append(item)
        return outList
#endDef

servicembean = AdminControl.queryNames("type=TransactionService,*" )
mbean = AdminControl.queryNames("type=Transaction,*" )
input = 0

while (input >= 0):
        tranList = wsadminToList(AdminControl.invoke(servicembean, "listManualTransactions" ))

        tranLength = len(tranList)
        if (tranLength > 0):
                print "----Manual Transaction details---------------"
                index = 0
                for tran in tranList:
                        print "   Index=" , index , " tran=" , tran
                        index = index+1
                #endFor
                print "----End of Manual Transactions ---------------"
                print "Select index of transaction to commit/rollback:"
                input = sys.stdin.readline().strip()
                if (input == ""):
                        print "No index selected, exiting."
                        input = -1
                else:
                        tran = tranList[int(input)]
                        commaPos = (tran.find(",") -1)
                        localTID = tran[0:commaPos+1]
                        print "Enter c to commit or r to rollback transaction ", localTID
                        input = sys.stdin.readline().strip()
                        if (input == "c"):
                                print "Committing transaction=", localTID
                                AdminControl.invoke(mbean, "commit", localTID )
                        #endIf
                        elif (input == "r"):
                                print "Rolling back transaction=", localTID
                                AdminControl.invoke(mbean, "rollback", localTID )
                        #endIf
                        else:
                                input = -1
                        #endelse
                #endElse
        else:
                print "No transactions found, exiting"
                input = -1
        #endElse
        print " "

#endWhile
```

Managing transaction logging for optimum server availability

You can manage transaction logging to optimize the availability of your application servers.

About this task

The transaction service writes information to the transaction log for every global transaction that involves two or more resources, or that is distributed across multiple servers. The transaction log is stored on disk and is used by the transaction service for recovery after a system or server crash. The transaction log for each application server consists of multiple subdirectories and files held in a single directory. To change the directory that an application server uses to store the transaction log, change the transaction log directory in the transaction service settings.

When a global transaction is completed, the information in the transaction log is no longer required, and the information is marked for deletion. The redundant information is garbage collected and intervals, and

the space is reused by new transactions. The log files are created with a fixed size at server startup, so no further disk space allocation is required during the lifetime of the server.

If all the log space is in use when a transaction needs to save information, the transaction is rolled back and the message `CWWTR0083W: The transaction log is full. Transaction rolled back.` is reported to the system error log. No more transactions can commit until more log space is made available when existing active transactions complete.

The default disk space allocation for the transaction logs is 1M. For global transactions that involve only XA resources and that are either local to an application server or are distributed between enterprise beans running in remote application servers, the default disk space allocation is suitable for peak workloads of up to 4000 concurrent two-phase commit transactions. For global transactions that involve Web Services Atomic Transaction (WS-AT) transactions or interoperable OTS transactions, the default disk space allocation is suitable for peak workloads of up to 250 concurrent two-phase commit transactions. For higher workloads, consider using a larger transaction log. To change the disk space allocation for the transaction log files, change the transaction log directory in the transaction service settings.

You can monitor the number of concurrent global transactions by using the performance monitoring counters for transactions. The “Global transaction commit time” counter is a measure of how long a transaction takes to complete and, therefore, how long the log is in use by a transaction. If this value is high, then transactions are taking a long time to complete, which can be due to resource manager or network failures. If you ensure that this value is low, the log is more efficiently used and unlikely to become full.

Use the following tasks to manage transaction logging to optimize the availability of your application servers:

Procedure

- “Configuring transaction aspects of servers for optimum availability.” Use this task to configure the transaction properties for an application server to help transactions to complete or recover more quickly.
- “Moving a transaction log from one server to another” on page 2513 Use this task if you have to move a transaction log between servers.
- “Restarting an application server on a different host” on page 2514 Use this task to restart the application server if you move a transaction log between hosts.
- “Configuring transaction properties for an application server” on page 2491 Use this task to change the directory or the disk space allocation for the transaction log files.

Configuring transaction aspects of servers for optimum availability

You can configure transaction-related aspects of application servers to optimize the availability of those servers. This helps your transactions to complete or recover more quickly. After changing transaction-related properties of an application server, you must restart the server.

About this task

To configure transaction-related aspects of application servers for optimum availability, complete the following steps:

Procedure

1. Store the transaction log files on a fast disk in a highly-available file system, such as a RAID device. The transaction log might have to be accessed by every global transaction and be used for transaction recovery after a crash. Therefore, the disk the log files are being written to should be on a highly-available file system, such as a RAID device.

   The performance of the disk also directly affects the transaction performance. In general, a global transaction makes two disk writes, one after the prepare phase when the outcome of the transaction is

known (this information is forced to disk) and a further disk write at transaction completion. Therefore, the transaction logs should be placed on the fastest disks available.

In order for automatic failover of transaction log recovery to work in a WebSphere Application Server cluster topology, network mounted devices must be used for the transaction logs, on a fast disk in a highly-available file system, such as a RAID device, that each cluster member can access.

2. Mirror the transaction log files by using hardware disk mirroring or dual-ported disks. If log files have been mirrored or can be recovered, these log files can be used when restarting a failed server, or they can be moved to another machine and another server can be started to undertake recovery.

   You can configure hardware disk mirroring or dual-ported disks by using the administrative console to specify the appropriate file system directory for the transaction logs.

3. Specify the optimum location of the transaction log directory for application servers.

   By default, an application server places transaction log files in a subdirectory of the installed WebSphere Application Server, where the subdirectory name is the same as the server name.

   For example, the default directory for an application server named `server1` is

   ```
   /QIBM/UserData/WebSphere/AppServer/was_version/Base/profiles/profile_name/tranlog/server1
   ```

   where *was_version* indicates the version number for this installation of IBM WebSphere Application Server. For example `V6` for WebSphere Application Server Version 6.

   You can define a specific location for the transaction log directory for an application server by setting the **Transaction Log Directory** property for the server. If the directory for the transaction logs has not been created at application server start up, the directory structure is created for you.

   Note: If you change the transaction log directory, apply the change and restart the application server as soon as possible, to minimize the risk of problems occurring before the application server is restarted. For example, if a problem causes the server to fail (with in-flight transactions), the server next starts with the new log directory and cannot automatically resolve in-flight transactions that were recorded in the old log directory.

4. Never allow more than one application server to concurrently use the same set of log files. Because the transaction logs record the state of global transactions within a server, if the logs become lost or corrupt, then transactions that are in the prepared state before failure can leave resources in an in-doubt state and prevent further updates or access to the resources by other users or servers. These transactions might have to be manually resolved by either committing or rolling back the transactions at the affected resource managers. The failed server can then be cold-started, which creates new empty transaction logs.

   If log files have been mirrored or can be recovered, these log files can be used when restarting a failed server, or they can be moved to another machine and another server can be started to undertake recovery, as described in the related tasks.

   Never allow more than one application server to concurrently use the same set of log files, because each server will destroy the information recorded by the other, resulting in corrupt log files that are unusable for future recovery purposes.

5. Configure application servers to always use the same listening port address at each startup. If you are running distributed transactions between multiple application servers, for example non-WebSphere EJB or Corba servers, the remote object references saved in the transaction log have to be redirected to the originating server on recovery.

   You must handle the redirection of remote object references so that transaction recovery can complete. For example, you must do this if an application server is deployed on WebSphere Application Server and runs distributed transactions with non-WebSphere EJB or Corba servers.

   In particular, the default restart action of an application server is to use a different listening port address to the port when the server shuts down. This prevents transaction recovery from completing. To overcome this, you must configure application servers to always use the same listening port address at each startup (see the ORB property com.ibm.CORBA.ListenerPort in the topic about Object

Request Broker custom properties). You might have to make similar configuration changes to other application servers involved in transactions, so that you can access those servers during recovery.

Moving a transaction log from one server to another

You can move the transaction logs for an application server to another server.

About this task

The following steps explain how to move transaction logs from one application server to another.

Note: The transaction service does not allow transaction logs to be moved from one server to another in a high availability (HA) environment. In such scenarios, WebSphere Application Server dynamically controls which server is assigned ownership of the recovery logs. For this reason, the following steps are not applicable when the **Enable failover of transaction log recovery** option has been set on the administrative console.

Procedure

1. Move all the transaction log files for the application server. The transaction log directory for each server contains a number of files and subdirectories. When moving transaction logs from one server to another you must move all of the files and subdirectories together as a single unit; otherwise recovery might not complete resulting in data inconsistency.
2. Follow the relevant step below, depending on your server configuration:
   a. Optional: For a server configuration where there are no distributed transactions, move the transaction logs to any server that has access to the same resource managers. For a single server or network-deployed server configuration where it is known there are no distributed transactions present in the logs, the transaction logs can be moved to any server (on any node) that has access to the same resource managers as the original server. For example, the server needs communication and valid security access to databases or message queues.

      If the server is in a different cell from the original server, you must ensure that there is a Java Authentication and Authorization Service (JAAS) alias available to the server that was used by the original server for accessing XA resources. In this case you should use wsadmin to construct the aliases, because if you use the administrative console to create the alias, then the node name gets prefixed to the alias.

      All the transaction log files for the original server must be moved to a directory accessible by the new server. This can be accomplished by either renaming the transaction log directory or copying all the contents to the new server transaction log directory before starting the new server.

   b. Optional: For a network-deployed server configuration where there are distributed transactions, move the transaction logs to a server that has the same name and host IP address, and access to the same resource managers. For a network-deployed server configuration, when it is known there are distributed transactions present in the logs, there are more restrictions. Distributed transactions that access multiple servers log information about each server involved in the transaction. This information includes the server name and the IP address of the machine on which the server is running. When recovery is taking place on server restart, the server uses this information to contact the distributed servers and similarly, the distributed servers try to contact the server with the same original name. Therefore, if a server fails and the logs have to be recovered on an alternative server, the alternative server must have the same name and host IP address as the original server. The alternative server also must have access to the same resource managers as the original server. For example, the server needs communication and valid security access to databases or message queues.

      Note: All servers within a cell must have unique names.

   Note: To complete transaction recovery, the application server uses the resource manager configuration information in the transaction logs. However, for the application server to continue

to do new work with the same resource managers, the server must have an appropriate resource manager configuration (as for the original server).

Restarting an application server on a different host

As part of restarting an application server on a different host, you move all the transaction logs to the new host.

About this task

Moving transactions logs to a different host is similar to moving logs from one server to another, as described in “Moving a transaction log from one server to another” on page 2513.

This involves moving an original application server on one host to an alternative server, which has access to the same resource managers, on another host. For a network-deployed server configuration, the alternative server must have the same name and host IP address as the original server.

Note: To complete transaction recovery, the application server uses the resource manager configuration information in the transaction logs. However, for the application server to continue to do new work with the same resource managers, the server must have an appropriate resource manager configuration (as for the original server).

To restart an application server on a different host, complete the following steps:

Procedure

1. Ensure that the alternative application server is stopped.
2. Move all the transaction logs for the original server to the alternative application server, as described in “Moving a transaction log from one server to another” on page 2513.
3. Restart the alternative application server.

Displaying transaction recovery audit messages

You can choose whether information messages are displayed on the administrative console and written to the `SystemOut.log` file upon transaction service recovery. To do this, set the DISABLE_RECOVERY_AUDIT_LOGGING custom property for the transaction service for the server.

About this task

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

On distributed platforms the default is for information messages to appear both on the administrative console and in the `SystemOut.log` file during the recovery of transaction services. If you do not want these messages to be displayed you can use the DISABLE_RECOVERY_AUDIT_LOGGING custom property.

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Transaction Service** > **[Additional Properties] Custom Properties**.
2. Click **New**.
3. Enter the values required, depending on the operating system:

- To suppress the information messages, type DISABLE_RECOVERY_AUDIT_LOGGING in the **Name** field, and TRUE in the **Value** field.

4. Click **Apply** or **OK**.
5. Save your changes to the master configuration.
6. Restart the server in recovery mode.

Delaying the cancelling of transaction timeout alarms

If the before completion stage of a transaction process is likely to include processes that could either take a long time to complete or could fail, then you might want the transaction to time out.

About this task

By default, transaction timeout alarms are cancelled prior to the before completion phase of the transaction begins. The DELAY_CANCELLING_ALARMS custom property allows the before completion phase of the transaction to be encompassed within the transaction timeout period. To do this, set the custom property on the application server.

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Transaction Service** > **[Additional Properties] Custom Properties**.
2. Click **New**.
3. Type DELAY_CANCELLING_ALARMS in the **Name** field, and TRUE in the **Value** field.
4. Click **Apply** or **OK**.
5. Save your changes to the master configuration.
6. Restart the server in recovery mode.

Removing entries from the transaction partner log

You can remove entries from the transaction partner log file. To do this, set the REMOVE_PARTNER_LOG_ENTRY custom property for the transaction service on the server that owns the partner log.

About this task

As part of the transaction recovery process, the partner log is checked to establish which resources are needed. If you want to remove certain entries from the partner log, such as a resource that no longer exists, set this custom property on the application server that owns the transaction partner log containing the entries you want to remove.

The REMOVE_PARTNER_LOG_ENTRY custom property is effective only when both of the following situations apply.

- The application server is started in recovery mode.
- The application server has no transactions that currently require recovery. You can establish this by checking the SystemOut.log file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using SystemOut.log , SystemErr.log, trace.log, and activity.log files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Transaction Service** > **[Additional Properties] Custom Properties**.
2. Click **New**.
3. Type `REMOVE_PARTNER_LOG_ENTRY` in the **Name** field, and in the **Value** field, type one or more comma-delimited integer recovery IDs to be removed.
4. Click **Apply** or **OK**.
5. Save your changes to the master configuration.
6. Restart the server in recovery mode.

Chapter 28. Administering web applications

This page provides a starting point for finding information about web applications, which are comprised of one or more related files that you can manage as a unit, including:

- HTML files
- Servlets can support dynamic web page content, provide database access, serve multiple clients at one time, and filter data.
- Java ServerPages (JSP) files enable the separation of the HTML code from the business logic in web pages.

IBM extensions to the JSP specification make it easy for HTML authors to add the power of Java technology to web pages, without being experts in Java programming. More introduction...

Deploying JavaServer Pages and JavaServer Faces files

JSP class loading settings

You can configure a JavaServer Pages (JSP) class to be loaded by either the JSP engine's class loader or by the web module's class loader.

By default, a JSP class is loaded by a unique instance of the JSP engine's class loader. The JSP engine's class loader enables reloading at runtime of a JSP class when the JSP source or one of its dependents is modified. This allows you to reload a single JSP class when necessary, without affecting any other loaded JSP classes.

JSP classes are loaded by the web module's class loader under either of the following scenarios.

1. The JSP engine configuration parameter useFullPackageNames is set to true, and the JSP file is configured as a servlet in the `web.xml` file using the <servlet-class> scenario in the table below.
2. The JSP engine configuration parameters useFullPackageNames and disableJspRuntimeCompilation are both set to true. In this case, you do not need to configure a JSP file does as a servlet in the `web.xml` file.

Configuring JSP files as Servlets

You can configure a JSP file as a servlet in the `web.xml` file. There are two ways to do this. They are described in the table below.

Before you configure a JSP file as a servlet, consider the following.

1. Reloading capability - If runtime reloading of JavaServer Pages files is desired, requests for JavaServer Pages files must be handled by the JSP engine. The <servlet-class> scenario in the table below disables runtime JSP file reloading, while the <jsp-file> scenario is compatible with reloading.
2. Reducing the number of class loaders - If you do not require runtime reloading of modified JSP pages and you want to reduce the number of class loader instances, then you can use the <servlet-class> scenario in the table below. Similarly, scenario 2 in section 1 above can be used without having to configure a JSP file as a servlet.

Table 236. Example: Configure a JSP file as a servlet in the `web.xml` file.. Configure a JSP file as a servlet

Scenario	Example	compatible with runtime reloading	multiple class loaders used?	useFullPackageNames

Table 236. Example: Configure a JSP file as a servlet in the web.xml file. (continued). Configure a JSP file as a servlet

<jsp-file>	<servlet> <servlet-name>jspOne</servlet-name> <jsp-file>jspOne.jsp</jsp-file> </servlet>	Yes	Yes	Can be true or false
<servlet-class>	<servlet> <servlet-name>jspTwo</servlet-name> <servlet-class>_ibmjsp.jspTwo</servlet-class> </servlet>	No	No	Must be true

The JSP batch compiler tool helps you configure JavaServer Pages files as servlets. When useFullPackageNames is true, the JSP batch compiler generates <servlet> and <servlet-mapping> elements for each JSP file that it successfully translates and compiles. The elements are written to a web.xml fragment file named generated_web.xml which is located in the binaries WEB-INF directory of a web module processed by the JSP file batch compiler (this directory is located within the deployed application's ear file). You can copy and paste all or some of these elements into the web.xml file to configure JavaServer Pages files as servlets.

Take note of the location of the web.xml that is used by the application server. The application specific configuration is obtained from either the application binaries (the application's ear file) or from the configuration repository. If an application is deployed into WebSphere Application Server with the flag Use Binary Configuration set to true, then the WEB-INF/web.xml file is looked for in a web module's binaries directory, not in the configuration repository. Below are examples of these two locations.

- An example of a configuration repository directory is *profile_root*/config/cells/*cellName*/applications/*enterpriseAppName*/deployments/*deployedName*/*webModuleName*
- An example of an application binaries directory is: *profile_root*/installedApps/*nodeName*/*EnterpriseAppName*/*WebModuleName*/

If the JSP batch compiler is executed on a pre-deployed application then the web.xml file is in the web module's WEB-INF directory.

JavaServer Pages (JSP) runtime reloading settings

JavaServer Pages files can be translated and compiled at run time when the JSP file or its dependencies are modified. This is known as JSP reloading.

Note: Use an assembly tool, such as Rational Application Developer, to modify IBM extension and binding files. You can convert extension and binding files within modules from XMI to XML using the IBM Bindings and Extensions Conversion Tool for Multi-Platforms.

JSP reloading is enabled through the **reloadEnabled** JSP engine parameter in the WEB-INF/ibm-web-ext.xmi or WEB-INF/ibm-web-ext.xml file.

ibm-web-ext.xmi example:

```
<jspAttributes xmi:id="JSPAttribute_1" name="reloadEnabled" value="true"/>
```

ibm-web-ext.xml example:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">
```

```
   <jsp-attribute name="trackDependencies" value="true" />
   <jsp-attribute name="disableJspRuntimeCompilation" value="true" />
   <jsp-attribute name="reloadEnabled" value="true"/>

   <reload-interval value="5"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="false"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named ibm-*-ext.xmi or ibm-*-bnd.xmi where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

The following table contains suggested reload settings for production and development environments.

Table 237. Suggested reload settings for production and development environments.. Reload settings

	Settings	
Configuration Attribute	**Production Environment**	**Development Environment**
reloadEnabled	false	true
reloadInterval	n/a (ignored if reloadEnabled is false)	approximately 5 seconds
trackDependencies	n/a (ignored if reloadEnabled is false)	true Alternatively, set this to false to improve response time if dependencies are not changing
disableJspRuntimeCompilation	true - Alternatively, set this to false if JSP files are not pre-compiled and therefore need to be compiled on the first request.	false

The default for the **reloadEnabled** parameter is true. If the **reloadEnabled** parameter is set to true, a JSP file is reloaded at run time if the JSP file and its class file do not have the same timestamp. In addition, if **trackDependencies** is set to true then the JSP file is reloaded if the timestamp of any of its dependencies has changed since the JSP class file was last generated. If the **reloadEnabled** parameter is set to false, a JSP file is still compiled if necessary on the first request to it unless the parameter **disableJspRuntimeCompilation** is true. For example, when **disableJspRuntimeCompilation** is false and **reloadEnabled** is false, a JSP file is compiled on the first request if the class file is outdated. It would not compile on subsequent requests, even if the JSP source file is modified or the class file is deleted,, unless **reloadEnabled** is true.

Reload interval

The reload interval is set through the **reloadInterval** JSP engine parameter.

`ibm-web-ext.xmi` example:

```
<jspAttributes xmi:id="JSPAttribute_1" name="reloadInterval" value="5"/>
```

ibm-web-ext.xml example:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">

   <jsp-attribute name="trackDependencies" value="true" />
   <jsp-attribute name="disableJspRuntimeCompilation" value="true" />
   <jsp-attribute name="reloadInterval" value="5"/>

   <reload-interval value="5"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="false"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

If reloading is enabled, the **reloadInterval** parameter value determines the delay between checks to see if a JSP file is outdated. For example, if **reloadInterval** is 5, the JSP engine checks to see if a JSP file is outdated only when the last such check was done more than five seconds prior to the current request for the JSP file. Once the **reloadInterval** is exceeded, reload checking is performed and the reload interval timer is reset to 0 for that JSP file. The larger the **reloadInterval**, the less frequently the JSP engine checks for the need to reload a JSP file.

Dependency tracking

Dependency tracking is set through the **trackDependencies** JSP engine parameter.

ibm-web-ext.xmi example:

```
<jspAttributes xmi:id="JSPAttribute_1" name="trackDependencies" value="true"/>
```

ibm-web-ext.xmi example:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">

   <jsp-attribute name="trackDependencies" value="true" />
   <jsp-attribute name="disableJspRuntimeCompilation" value="true" />
   <jsp-attribute name="reloadInterval" value="5"/>

   <reload-interval value="5"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="false"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

If reloading is enabled, the **trackDependencies** parameter value determines whether the JSP engine tracks modifications to the requested JSP file dependencies as well as to the JSP file itself. The three types of dependencies tracked by the JSP engine are:

- files statically included in the JSP file
- tag files that are referenced in the JSP file (excluding tag files that are in JAR files)
- TLDs that are referenced in the JSP file (excluding TLDs that are in JAR files)

Dependency tracking information is always included in the generated class file even if **trackDependencies** is false. The information is not used by the JSP engine or batch compiler unless the **trackDependencies** parameter is true. This means that you can enable dependency tracking without having to recompile JSP files.

For example, the toplevel.jsp file statically includes the footer.jspf file. When the toplevel.jsp file is compiled, the path to the footer.jspf file and its timestamp are stored in the toplevel.jsp's class file. As a result, the footer.jspf file is modified and the toplevel.jsp file is requested. Now that the reload interval for the toplevel.jsp file has been exceeded, the JSP engine compares the timestamp stored in the class file with the footer.jspf file timestamp on disk. Because the timestamps are different, the toplevel.jsp file is compiled, picking up the modification to the footer.jspf file. In order for dependency tracking to work, the **trackDependencies** value must be set to true at the time a JSP file is requested at run time or is processed by the batch compiler.

Disabling compilation

Disablement of run time compilation of JavaServer Pages is set via the disableJspRuntimeCompilation JSP engine parameter.

ibm-web-ext.xmi example:

```
<jspAttributes xmi:id="JSPAttribute_1" name="disableJspRuntimeCompilation" value="true"/>
```

ibm-web-ext.xml example:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">

   <jsp-attribute name="trackDependencies" value="true" />
   <jsp-attribute name="disableJspRuntimeCompilation" value="true" />
   <jsp-attribute name="reloadInterval" value="5"/>

   <reload-interval value="5"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="false"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

If the **disableJspRuntimeCompilation** parameter is set to true, the JSP engine at run time does not translate and compile JSP files; the JSP engine loads only precompiled class files. JSP source files do not need to be present in order for the class files to be loaded. With this option set to true, an application can be installed without JSP source, but must have precompiled class files. There is a web container custom property of the same name that can be used to determine the behavior of all web modules installed in a server. If both the web container custom property and the JSP engine option are set, the JSP engine option takes precedence. Setting the **disableJspRuntimeCompilation** parameter to true automatically sets **reloadEnabled** to false.

Reload processing sequence

The processing sequence pertaining to JSP file reloading when **trackDependencies** is false is shown in Figure 1.

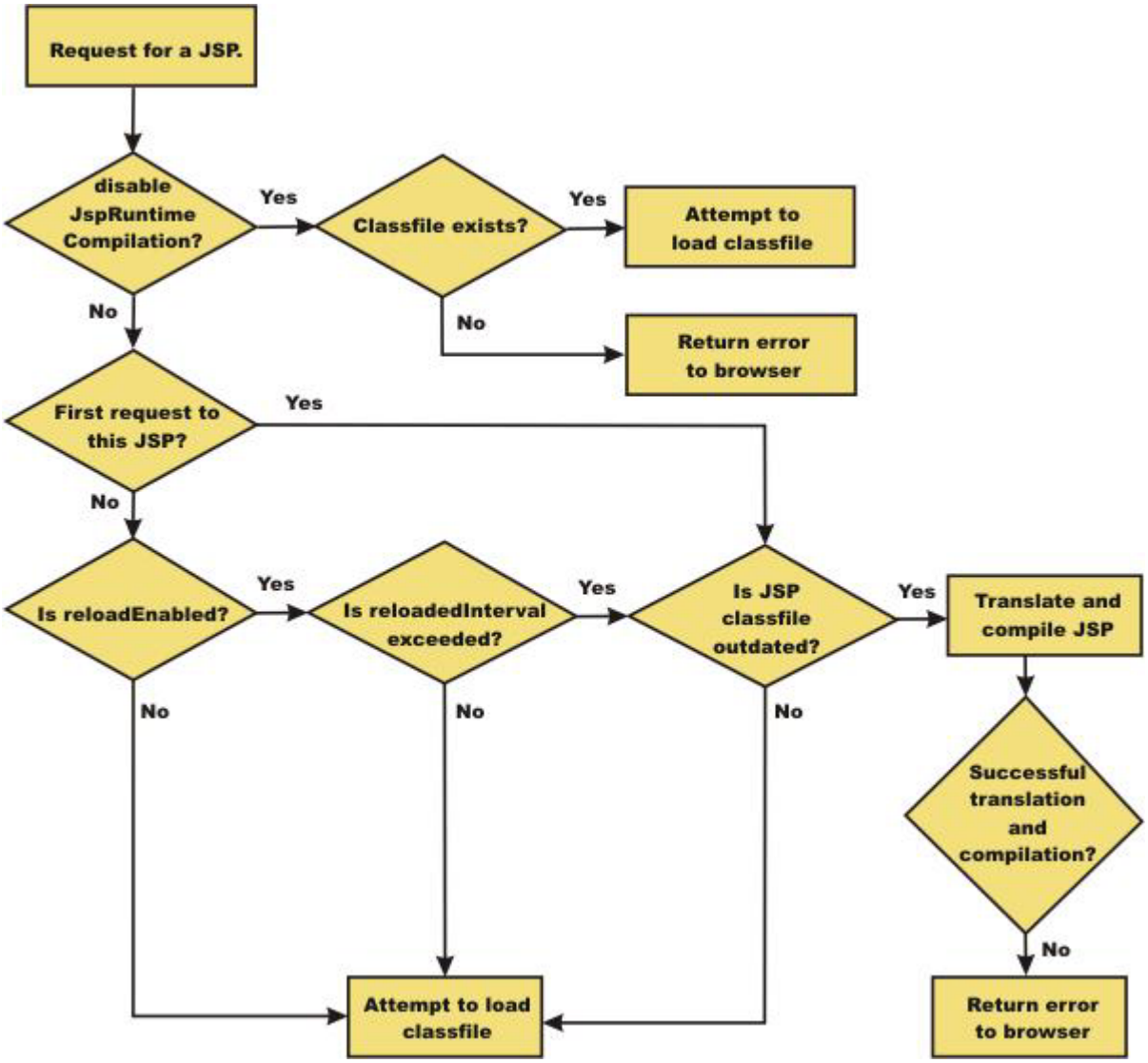


Figure 40. Reload processing sequence when **trackDependencies** *is false.*

When **trackDependencies** is true, the JSP engine does additional file system processing to determine if any of a JSP file's dependencies have changed since the JSP file was last translated and compiled. Figure 2 shows the additional processes that are performed on the 'No' path of flow chart labeled "is JSP class file outdated?". You can see that the path taken when disableJspRuntimeCompilation is true is the most efficient path.

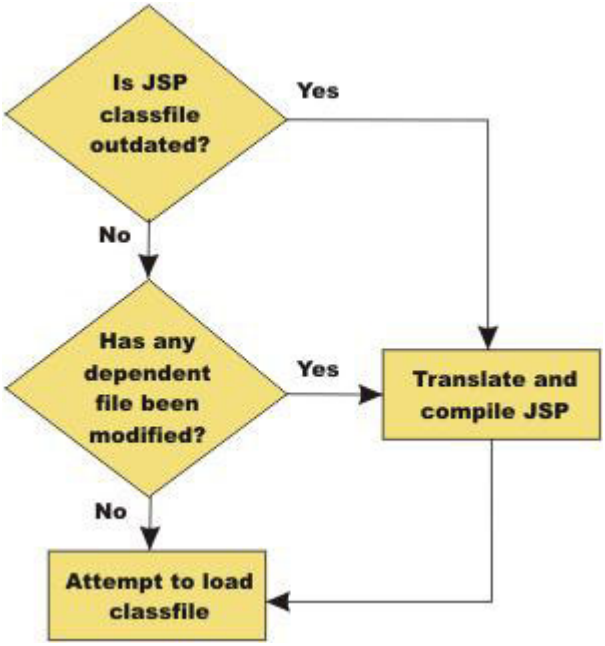


Figure 41. Additional reload processing performed when **trackDependencies** *is true.*

JSP and JSF option settings

Use this panel to configure the class reloading of web modules such as JavaServer Pages (JSP) files and to select a JSF implementation to use with this application.

To view this administrative console panel, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **JSP and JSF options**. This panel is the same as the **Provide JSP reloading options for web modules** panel on the application installation and update wizards.

The following note applies to the files with a .xmi extension in this topic:

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named ibm-*-ext.xmi or ibm-*-bnd.xmi where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

Web module

Specifies the name of a web module in the installed or deployed application.

URI
Specifies the location of the module that is relative to the root of the application (EAR file).

JSP enable class reloading
Specifies whether to enable class reloading when JSP files are updated.

A web container reloads JSP files only when the IBM extension reloadEnabled in the jspAttributes of the `ibm-web-ext.xmi` file is set to `true`.

Java Platform, Enterprise Edition 5 (Java EE 5) applications IBM extension files are in `.xml` file format. For applications versions earlier than Java EE 5, they are in the `.xmi` file format.

JSP reload interval in seconds
Specifies the number of seconds to scan the application file system for updated JSP files. The default is the value of the reloading interval attribute in the IBM extension (`META-INF/ibm-web-ext.xmi`) file of the web module.

To enable reloading, specify a value greater than zero (for example, 1 to 2147483647). The default reload interval is 5. To disable reloading, specify zero (0). The range is from 0 to 2147483647.

The reloading interval attribute takes effect only if class reloading is enabled.

Java EE 5 applications IBM extension files are in `.xml` file format. For applications versions earlier than Java EE 5, they are in the `.xmi` file format.

Sun Reference Implementation 1.2
Select this option to use the Sun Reference Implementation 1.2 JSF implementation.

If you change the JSF implementation that you are using for your application, you must delete any previously compiled JSP files. If you precompiled your application, you must recompile. If you did not precompile, but have already requested JSP files from this application, you must delete the JSP files from the temp directory of your profile.

You can set the JSF engine configuration parameter, com.ibm.ws.jsf.JSF_IMPL_CHECK, to true to automatically mark the JSP files to recompile at application startup.

MyFaces 2.0
Select this option to use the MyFaces JSF implementation. This is the default JSF implementation.

If you change the JSF implementation that you are using for your application, you must delete any previously compiled JSP files. If you precompiled your application, you must recompile. If you did not precompile, but have already requested JSP files from this application, you must delete the JSP files from the temp directory of your profile.

You can set the JSF engine configuration parameter, com.ibm.ws.jsf.JSF_IMPL_CHECK, to true to automatically mark the JSP files to recompile at application startup.

In a mixed-version cell, a V7 node uses MyFaces 1.2 if the MyFaces selection is toggled, while a V8 node uses MyFaces 2.0. For WebSphere Application Server versions before V7 (for example, V6.1 and earlier), this toggle is ineffective because JSF implementation switching was not supported before V7.

JSP run time compilation settings
By default, the JavaServer Pages (JSP) engine translates a requested JSP file, compiles the `.java` file, and loads the compiled servlet into the run time environment. You can change the JSP engine default behavior by indicating that a JSP file must not be translated or compiled at run time, even when a `.class` file does not exist.

If run time compilation is disabled, you must precompile the JSP files, which provides the following advantages:
- Reduces compilation related disk operations.
- Minimizes disk storage requirements necessary for handling temporary `.java` files generated during a run time compilation.
- Allows you to not include the JSP source files in the application.
- Allows verification that a JSP file compiled successfully before deploying and installing the application in the product

You can disable run time JSP file compilation on a global or an individual web application basis:
- To disable the translation and compilation of JSP files for all Web applications, in the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name*. Then, in the Container Settings section, click **Web container settings > Web container > Custom properties**.

  If the disableJspRuntimeCompilation property appears in the list of defined custom properties, but is set to `false`, click the property name, and then set the property to `true`.

  If this property is not included in the list of defined custom properties, click **New**, and then specify `disableJspRuntimeCompilation` in the **Name** field and `true` in the **Value** field.

  Valid settings for this property are `true` or `false`. When this property is set to `true`, translation and compilation of the JSP files is disabled at run time for all web applications.
- To disable the translation and compilation of JSP files for a specific web application, set the JSP engine initialization parameter disableJspRuntimeCompilation to `true`. This setting, if enabled, determines the run time behavior of the JSP engine and overrides the web container custom property setting.

  Set this parameter through the **JavaServer Pages attribute assembly settings** page when assembling applications.

  Valid values for this setting are `true` or `false`. If this parameter is set to `true`, then, for that specific web application, translation and compilation of the JSP files is disabled at run time, and the JSP engine only loads precompiled files.
- If neither the web container custom property nor the JSP parameter is set, the first request for a JSP file results in the translation and compilation of the JSP file when the `.class` file does not exist or is outdated. Subsequent requests for the file also result in translations and compilations, but only if the following conditions are met:
  - Translations are required because the `.class` file is outdated.
  - Reloading is enabled for the web module.
  - Reload interval is exceeded.

If you disable run time compilation and a request arrives for a JSP file that does not have a matching .class file, the JSP engine returns the following 404 error to the browser:

```
Error 404: SRVE0200E: Servlet [org.apache.jsp._jsp1]: Could not find required servlet class - _jsp1.class
```

In this case, an exception is written to the System Out (SYSOUT) and First Failure Data Capture (FFDC) logs. .

If a JSP file has a matching .class file but that file is out of date, the JSP engine still loads the .class file into memory.

Provide options to compile JavaServer Pages settings

Use this page to specify options to be used by the JavaServer Pages (JSP) compiler.

This administrative console page is a step in the application installation and update wizards. To view this page, you must select **Precompile JavaServer Pages files** on the **Select installations options** page. Thus, to view this page, click **Applications** > **New Application** > **New Enterprise Application** > ***application_path*** > **Next** > **Detailed - Show me all installation options and parameters** > **Next** > **Next** or **Continue** > **Precompile JavaServer Pages files** > **Next** > **Step: Provide options to compile JSPs**.

You can specify the JSP compiler options on this page only when installing or updating an application that contains web modules. After the application is installed, you must edit the JSP engine configuration parameters of a web module `WEB-INF/ibm-web-ext.xmi` file to change its JSP compiler options.

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named ibm-*-ext.xmi or ibm-*-bnd.xmi where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

Web module
Specifies the name of a module within the application.

URI
Specifies the location of the module relative to the root of the application (EAR file).

JSP class path
Specifies a temporary class path for the JSP compiler to use when compiling JSP files during application installation. This class path is not saved when the application installation is complete and is not used when the application is running. This class path is used only to identify resources outside of the application which are necessary for JSP compilation and which will be identified by other means (such as shared libraries) after the application is installed. In network deployment configurations, this class path is specific to the deployment manager machine.

To specify that multiple web modules use the same class path:
1. In the list of web modules, select the **Select** check box beside each web module that you want to use a particular class path.
2. Expand **Apply Multiple Mappings**.
3. Specify the desired class path.
4. Click **Apply**.

Use full package names
Specifies whether the JSP engine generates and loads JSP classes using full package names.

When full package names are used, precompiled JSP class files can be configured as servlets in the `web.xml` file, without having to use the `jsp-file` attribute. When full package names are not used, all JSP classes are generated in the same package, which has the benefit of smaller file-system paths.

When the options useFullPackageNames and disableJspRuntimeCompilation are both `true`, a single class loader is used to load all JSP classes, even if the JSP files are not configured as servlets in the `web.xml` file.

This option is the same as the useFullPackageNames JSP engine parameter.

JDK source level
Specifies the source level at which the Java compiler compiles JSP Java sources. Valid values are `13`, `14`, and `15`. The default value is `13` for pre-Java EE 5 web modules, which specifies source level 1.3 and `15` for Java EE 5 and later web modules.

Disable JSP runtime compilation
Specifies whether a JSP file should never be translated or compiled at run time, even when a `.class` file does not exist.

When this option is set to `true`, the JSP engine does not translate and compile JSP files at run time; the JSP engine loads only precompiled class files. JSP source files do not need to be present in order to load class files. You can install an application without JSP source, but the application must have precompiled class files.

For a single web application class loader to load all JSP classes, this compiler option and the **Use full package names** option both must be set to `true`.

This option is the same as the disableJspRuntimeCompilation JSP engine parameter.

Administering web applications

Modifying the default web container configuration
A web container handles requests for servlets, JavaServer Pages (JSP) files, and other types of files that include server-side code. The web container creates servlet instances, loads and unloads servlets, creates and manages request and response objects, and performs other servlet management tasks. The web server plug-ins, provided by the product, help supported web servers to pass servlet requests to web containers.

About this task

If the property to start servlets during application server startup is enabled, part of its startup process calls the Servlet.init method on its servlets when you start the web container. Therefore, when the web container starts and calls the init method, other components such as Naming and Work Load Management might not be fully started yet. As a result, application server related calls may not work because all of the application server components might not be ready yet. Once the application server is 'ready for e-business', it is completely ready. If application server related calls fail during Servlet.init method, you can either:

- Start the servlet manually when the server is ready for e-business instead of starting the servlet upon startup or
- You can choose not to make application server related calls in the servlet's init method.

The web container is created initially with default properties values suitable for simple web applications. However, these values might not be appropriate for more complex web applications.

Your application is considered complex if it requires any of the following features:
- Additional virtual host aliases
- Servlet caching
- Persistent HTTP session support
- Session tracking support with URL rewriting
- Special web container transport chain settings
- Asynchronous or remote dispatching
- No request or response pooling

Make the following configuration changes if you have a complex application:

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. Then under web container settings, click on one of the following:
   a. **Web container**, if your web application requires a virtual host, other than the default_host, or requires servlet caching.
   b. **Web container transport chains**, if you need to reconfigure your HTTP connections.
2. If your application handles special client request loads, in the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. Then under Additional Properties, click **Thread pools** to modify your thread pool settings.
3. If your application requires global settings for internal servlets for web application archive (WAR) files packaged by third-party tools, in the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Web Container Settings** > **Web container**. Then under Additional Properties, click **Custom properties** and enter the appropriate custom property.

Web container settings

Use this page to configure the web container settings.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Web Container Settings** > **Web container**.

Default virtual host:

Specifies a virtual host that enables a single host machine to resemble multiple host machines. Resources associated with one virtual host cannot share data with resources associated with another virtual host, even if the virtual hosts share the same physical machine.

Select a virtual host option:

default_host
The product provides a default virtual host with some common aliases such as the machine IP address, short host name, and fully qualified host name. The alias comprises the first part of the path for accessing a resource such as a servlet. For example, it is `localhost:9080` in the request `http://localhost:9080/myServlet`.

admin_host
This virtual host option is another name for the application server; also known as *server1* in the base installation. This process supports the use of the administrative console.

proxy_host
The virtual host called proxy_host, includes default port definitions, port 80 and 443, which are typically initialized as part of the proxy server initialization. Use this proxy host as appropriate with routing rules associated with the proxy server.

Enable servlet caching:

Specifies that if a servlet is started once and it generates output to be cached, a cache entry is created containing not only the output, but also side effects of the invocation. These side effects can include calls to other servlets or JavaServer Pages (JSP) files, as well as metadata about the entry, including timeout and entry priority information.

Portlet fragment caching requires that servlet caching is enabled. Therefore, enabling portlet fragment caching automatically enables servlet caching. Disabling servlet caching automatically disables portlet fragment caching.

Disable servlet request and response pooling:

Specifies to disable the pooling of servlet request and servlet response objects that are pooled by the web container. When you disable pooling of servlet request and servlet response objects, new servlet request and servlet response objects are created for each request.

When you disable pooling of servlet request and servlet response objects, new servlet request and servlet response objects are created for each request, which can negatively affect performance, but provides protection from any unforeseen pooling bugs.

Number of timeout threads:

Specifies the number of threads that are available to handle asynchronous servlet timeout operations per server.

The default of two might be too low if you have many applications using asynchronous servlets that often have timeouts.

Default timeout:

Specifies the default asynchronous servlet timeout for the server.

The default of 30 seconds can be lowered if responses are not being received quickly enough and there is a viable fall back in the error case. You can raise the value if too many timeouts are being received and the longer timeout produces responses in an acceptable manner to the client. The units are in millisecond, so multiply the number by 1000 to convert to seconds. To configure at a higher granularity, you must use the AsyncContext setTimeout method programmatically.

Use thread pool to start Runnable objects:

Select this option to use the same thread pool where the request originates. This option does not propagate any context from the original request.

Use a work manager to start Runnable objects:

Select this option to use an Asynchronous Beans work manager to start the runnable. This option is the default selection. The work manager option propagates any context that is configured for the selected work manager. This option also requires selecting the JNDI name of the work manager that you will use.

Considerations when using a work manager:

- The context that is propagated is configurable under Resources > Asynchronous Beans > Work managers. You can also create new work managers on the same panel.
- The WebSphere Application Server default Work Manger is used unless you specify otherwise. This might not be desirable as other components might be using the work manager and effectively decreasing the number of threads that are available at one time.
- To make changes to the work manager settings, it is recommended that you create a work manager so changes to defaults do not affect other components.
- If you have a work manager configured to throw exceptions when the work queue is full, then an exception of type IllegalStateException is thrown to the caller of start(Runnable) and the caller is responsible for handling the exception.

Programmatic session cookie configuration collection

Use this page to secure cookies by prohibiting programmatic configuration.

To view this administrative console page at the web container level, click **Security > Global security > Programmatic session cookie configuration**.

Cookie domain:

Specifies the domain where the cookie is used. Use the asterisk (*) symbol as a wild card, for example, *.ibm.com.

Cookie name:

Specifies an alphanumeric name for the cookie, for example, **JSESSIONID**. Use the asterisk (*) symbol as a wild card.

Cookie path:

Specifies the directory in the domain where the cookie is valid; for example, **/path_name**. Use the asterisk (*) symbol as a wild card.

Web container custom properties

You can configure name-value pairs of data, where the name is a property key and the value is a string value that you can use to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available in the administrative console. The following is a list of some of the available web container custom properties.

To specify web container custom properties:

1. In the administrative console click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Web Container Settings** > **Web container**.
2. Under **Additional Properties** select **Custom Properties**.
3. On the Custom Properties page, click **New**.
4. On the settings page, enter the name of the custom property that you want to configure in the **Name** field and the value that you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

The following is a list of custom properties provided with the Application Server. The topics, JavaServer Pages specific web container custom properties and HTTP transport custom properties, are listed in a separate topic.

You can use the Custom properties page to define the following properties for use by the Java virtual machine.

- “BodyContentBuffSize” on page 2532
- com.ibm.ws.jsf.disablealternatefacesconfigsearch
- “com.ibm.ws.jsp.enableDefaultIsELIgnoredInTag” on page 2532
- “com.ibm.ws.jsp.getWriterOnEmptyBuffer” on page 2532
- “com.ibm.ws.jsp.limitBuffer” on page 2533
- “com.ibm.ws.jsp.throwExceptionForAddELResolver” on page 2533
- “com.ibm.ws.webcontainer.assumefiltersuccessonsecurityerror” on page 2533
- “com.ibm.ws.webcontainer.channelwritetype” on page 2533
- “com.ibm.ws.webcontainer.checkEDRinGetRealPath” on page 2534
- “com.ibm.ws.webcontainer.copyattributeskeyset” on page 2534
- “com.ibm.ws.webcontainer.disableSetCharacterEncodingAfterParametersRead” on page 2534
- “com.ibm.ws.webcontainer.disableSystemAppGlobalListenerLoading” on page 2535
- “com.ibm.ws.webcontainer.disablexPoweredBy” on page 2535

- “com.ibm.ws.webcontainer.disallowAllFileServing” on page 2535
- “com.ibm.ws.webcontainer.disallowserveservletsbyclassname” on page 2535
- “com.ibm.ws.webcontainer.discernUnavailableServlet” on page 2535
- “com.ibm.ws.webcontainer.dispatcherRethrowSER” on page 2536
- “com.ibm.ws.webcontainer.dispatcherRethrowSError” on page 2536
- “com.ibm.ws.webcontainer.donotservebyclassname” on page 2536
- “com.ibm.ws.webcontainer.enabledefaultservletrequestpathelements” on page 2536
- “com.ibm.ws.webcontainer.enableErrorExceptionTypeFirst” on page 2537
- “com.ibm.ws.webcontainer.enableJspMappingOverride” on page 2537
- “com.ibm.ws.webcontainer.enableMultiReadOfPostData” on page 2537
- “com.ibm.ws.webcontainer.extractHostHeaderPort and trusthostheaderport” on page 2537
- “com.ibm.ws.webcontainer.finishresponseonclose” on page 2538
- “com.ibm.ws.webcontainer.ForceDifferentCookiePaths” on page 2538
- “com.ibm.ws.webcontainer.HTTPOnlyCookies” on page 2538
- “com.ibm.ws.webcontainer.ignoreinjectionfailure” on page 2539
- “com.ibm.ws.webcontainer.ignoreInvalidQueryString” on page 2539
- “com.ibm.ws.webcontainer.IgnoreSessiononStaticFileRequest” on page 2539
- “com.ibm.ws.webcontainer.invokeFilterInitAtStartup” on page 2539
- “com.ibm.ws.webcontainer.invokeFiltersCompatibility” on page 2539
- “com.ibm.ws.webcontainer.invokerequestlistenerforfilter” on page 2540
- “com.ibm.ws.webcontainer.KeepUnreadPostDataAfterResponseSentToClient” on page 2540
- “com.ibm.ws.webcontainer.logServletContainerInitializerClassloadingErrors” on page 2540
- “com.ibm.ws.webcontainer.mapFiltersToAsterisk” on page 2540
- “com.ibm.ws.webcontainer.normalizerequesturi” on page 2541
- “com.ibm.ws.webcontainer.parseUTF8PostData” on page 2541
- “com.ibm.ws.webcontainer.provideQStringToWelcomeFile” on page 2541
- “com.ibm.ws.webcontainer.SendResponseToClientAsPartOfSendRedirect” on page 2542
- “com.ibm.ws.webcontainer.SendResponseToClientWhenResponseIsComplete” on page 2542
- “com.ibm.ws.webcontainer.ServeWelcomeFileFromExtendedDocumentRoot” on page 2542
- “com.ibm.ws.webcontainer.ServletDestroyWaitTime” on page 2542
- “com.ibm.ws.webcontainer.setUnencodedHTMLinsendError” on page 2542
- “com.ibm.ws.webcontainer.suppressheadersinrequest” on page 2543
- “com.ibm.ws.webcontainer.suppressHtmlRecursiveErrorOutput” on page 2543
- “com.ibm.ws.webcontainer.suppressLastZeroBytePackage” on page 2543
- “com.ibm.ws.webcontainer.suppressServletExceptionLogging” on page 2543
- “com.ibm.ws.webcontainer.throwMissingJspException” on page 2544
- “com.ibm.ws.webcontainer.webgroupvhostnotfound” on page 2544
- “com.ibm.ws.webcontainer.xPoweredBy” on page 2544
- “com.ibm.wsspi.jsp.convertAttrValueToString” on page 2544
- “com.ibm.wsspi.jsp.disableElCache” on page 2544
- “com.ibm.wsspi.jsp.disableResourceInjection” on page 2544
- “com.ibm.wsspi.jsp.disableTldSearch” on page 2545
- “com.ibm.wsspi.jsp.enabledoublequotesdecoding” on page 2545
- “DebugSessionCrossover” on page 2545
- “DecodeUrlAsUTF8” on page 2546

- “enableInProcessConnections” on page 2546
- “fileServingEnabled, directoryBrowsingEnabled, and serveServletsByClassnameEnabled” on page 2546
- “ForceSessionIdLengthCheck” on page 2547
- “ForceSessionInvalidationMultiple” on page 2547
- “httpsIndicatorHeader” on page 2547
- “HttpSessionIdReuse” on page 2548
- “listeners” on page 2548
- “prependSlashToResource” on page 2548
- “trusted” on page 2549

BodyContentBuffSize: The size of the body content buffer for a JavaServer Pages (JSP) file can affect the performance of some applications. By default, the body content buffer size is 512 bytes. However, you can use the BodyContentBuffSize custom property to set a different buffer value.

Name	Default value
BodyContentBuffSize	512

com.ibm.ws.jsf.disablealternatefacesconfigsearch: Disables MyFaces searching for META-INF/*.faces-config.xml for all web applications on a server.

Name	Default value
com.ibm.ws.jsf.disablealternatefacesconfigsearch	false

com.ibm.ws.jsp.enableDefaultIsELIgnoredInTag: Typically Expression Language (EL) expressions in tag files get evaluated before the tag files a JavaServer Page (JSP) is compiled. However, under certain conditions these EL expressions in a tag file do not get evaluated if the <el-ignored> attribute is set to true.

To ensure that EL expressions are always evaluated, set the com.ibm.ws.jsp.enableDefaultIsELIgnoredInTag custom property to `true`. The default value for this property is false.

Name	Default value
com.ibm.ws.jsp.enableDefaultIsELIgnoredInTag	false

com.ibm.ws.jsp.getWriterOnEmptyBuffer: The dynamic cache service uses flushes to determine when one cacheable web fragment, such as a JSP include or a c:import, ends and the next web fragment begins. If you set the com.ibm.wsspi.jsp.usecdatatrim custom property to `true` for your JSP engine, all of the white space and extra lines in the generated Java code are stripped out. In this situation, there might not be any content to write before the first flush. If the generated Java code contains text or other code before the first flush then normal dynamic cache service processing occurs.

If you set the com.ibm.wsspi.jsp.usecdatatrim custom property to `true`, and are using the dynamic cache service, you must also set the com.ibm.ws.jsp.getWriterOnEmptyBuffer custom property to `true`. This custom property requires the JSP Engine to call the flush function when it reaches the end of the first cachable web fragments even if there is not any data to flush. The default value for this property is false.

Name	Default value
com.ibm.ws.jsp.getWriterOnEmptyBuffer	false

com.ibm.ws.jsp.limitBuffer: The body content buffer size of the tag bodies for a JavaServer Pages (JSP) file are reused to optimize performance. If the size of a tag body increases beyond the default body content buffer size, the buffer is resized to accommodate the tag body. However, the buffer is not reset to the default size after serving a request. As a result, the heap memory that is used by org.apache.jasper.runtime.BodyContentImpl implementation might increase over time. You can configure the body content buffer size by setting an integer value for the BodyContentBuffSize custom property. For more information, see "BodyContentBuffSize" on page 2532.

Use the com.ibm.ws.jsp.limitBuffer custom property to deallocate large body content buffer sizes and create a buffer with the default buffer size.

Name	Default value
com.ibm.ws.jsp.limitBuffer	false

com.ibm.ws.jsp.throwExceptionForAddELResolver: Set the com.ibm.ws.jsp.throwExceptionForAddELResolver property to `true` if you do not want to allow an ELResolver to be registered from a servlet or a filter after the application has received a request from the client. When this property is set to true, an IllegalStateException is thrown as specified by the JSP (Java Server Pages) specification for addELResolver() method of the JspApplicationContext interface.

The default value for this property is false.

Name	Default value
com.ibm.ws.jsp.throwExceptionForAddELResolver	false

com.ibm.ws.webcontainer.assumefiltersuccessonsecurityerror: When a request is received for a static file which does not exist, the web container calls defined servlet filters. If the filters do no successfully complete, a 404 error code is set. In a situation where application security is enabled, a security check is performed as part of filter invocation. Typically if the security check fails the web container considers the filters to have failed and still sets a 404 error code instead of the 401 error code that indicates the failure of a security check. The 404 error code enables the requester to access the static file without logging on.

You can set the com.ibm.ws.webcontainer.assumefiltersuccessonsecurityerror custom property to `true`, to prevent the 401 error code from being replaced with a 404 error code, and ensure that a user must enter a valid user ID and password before they can access a static file.

Name	Default value
com.ibm.ws.webcontainer.assumefiltersuccessonsecurityerror	false

com.ibm.ws.webcontainer.channelwritetype:

By default, the web container uses asynchronous writes to write response data in chunks up to the response buffer size. For larger responses that are greater than the response buffer size, the web container continues to buffer response data into memory while waiting for an asynchronous write of a response data chunk to complete. This process can result in part of a large response held in memory, which can lead to high memory usage and potentially an out of memory error. An application server hang might also occur when a server is simultaneously processing more requests than web container-defined threads.

If the com.ibm.ws.webcontainer.channelwritetype property is set to `sync`, synchronous writing is used, otherwise asynchronous writing is used by default. With synchronous writing, response data are written synchronously in chunks of up to the value of responsebuffersize and no response data are buffered into memory while waiting for a synchronous write of a response data chunk to complete. As a result, the approximate maximum amount of response data that is held in memory is equal to the responsebuffersize

multiplied by the number of web container threads. The maximum number of requests that can be processed simultaneously by the web container is limited by the number of web container threads. Additional requests are queued, waiting for a request that is in process to complete.

The responsebuffersize web container custom property defines the maximum amount of response data written by the web container in a single chunk, and is 32k by default. As a result, it is used to change the number of writes needed by the web container to send complete response data. However, if an application flushes response data, any response data held by the web container is immediately written irrespective of the responsebuffersize.

Use the following name-value pair to write chunks of data using synchronous writes.

Name	Default value
com.ibm.ws.webcontainer.channelwritetype	none

com.ibm.ws.webcontainer.checkEDRinGetRealPath: The ServletContext.getRealPath() Java Servlet API does not return the correct path for a requested resource when the resource exists in an extendedDocumentRoot path and does not exist in the installed application path. If you want the ServletContext.getRealPath() Java Servlet API to look for the requested resource in the extendedDocumentRoot path if the resource is not found in the installed application path, set the com.ibm.ws.webcontainer.checkEDRinGetRealPath custom property to `true`.

When this property is set to true, and the requested resource is also not found in the extendedDocumentRoot path, a null value is returned.

Name	Default value
com.ibm.ws.webcontainer.checkEDRinGetRealPath	false

com.ibm.ws.webcontainer.copyattributeskeyset: This custom property addresses a situation where the request.getAttributeNames method returns a list of values. If a servlet modifies the list using the request.removeAttribute method, subsequent calls to the nextElement method causes a java.util.ConcurrentModificationException exception. To enable a servlet to modify the list, set the `com.ibm.ws.webcontainer.copyattributeskeyset` custom property to true. When you set this custom property to true, a copy of the list of attributes is returned, which enables the servlet to modify the list without resulting in a java.util.ConcurrentModificationException exception when the nextElement method is called.

Name	Default value
com.ibm.ws.webcontainer.copyattributeskeyset	false

com.ibm.ws.webcontainer.disableSetCharacterEncodingAfterParametersRead: The web container processes a setCharacterEncoding(String) method of the ServletRequest API even if it is called after the post data is parsed. According to the Java Servlet Specification, the web container should ignore a setCharacterEncoding(String) method if the method is called after the data is parsed.

If you want the web container to ignore a setCharacterEncoding(String) method if the method is called after the data is parsed, add the com.ibm.ws.webcontainer.disableSetCharacterEncodingAfterParametersRead custom property to your web container configuration settings and set this property to `true`.

The default value for this property is false.

Name	Default value
com.ibm.ws.webcontainer.disableSetCharacterEncodingAfterParametersRead	false

com.ibm.ws.webcontainer.disableSystemAppGlobalListenerLoading:

If a system application is the first to start, and the application attempts to load a global listener in a shared library that is associated with the server classloader, the application does not load that listener and prevents the listener from being loaded or invoked by a later non-system application. Set the com.ibm.ws.webcontainer.disableSystemAppGlobalListenerLoading custom property to true to prevent system applications from loading global listeners. When this property is set to `true`, the system application does not attempt to load the global listeners and later non-system applications can load them from a shared library associated with a server class loader.

Name	Default value
com.ibm.ws.webcontainer.disableSystemAppGlobalListenerLoading	false

com.ibm.ws.webcontainer.disablexPoweredBy:

Note: When you configure server security, you can turn off the X-Powered-By header if you do not want to reveal which server you are running. Use this custom property to disable the X-Powered-By header, which prevents the header from being sent on the HTTP response. The default value is false. However, set this property to `true`, if you want to disable this header.

Name	Default value
com.ibm.ws.webcontainer.disablexPoweredBy	false

com.ibm.ws.webcontainer.disallowAllFileServing:

Use the com.ibm.ws.webcontainer.disallowAllFileServing custom property to disable file serving on all applications on a specific application server.

You can enable file serving on a global level across a given application server by using the fileServingEnabled custom property. However, the fileServingEnabled property is overridden by the specific deployment information of each application. Therefore, the current fileServingEnabled custom property only applies as a backup in case an application does not define the fileServingEnabled setting itself.

To globally override this setting on a specific application server to prevent the application server from serving static files regardless of their individual deployment settings, set the com.ibm.ws.webcontainer.disallowAllFileServing web container custom property to `true` using the following name-value pair.

Name	Default value
com.ibm.ws.webcontainer.disallowAllFileServing	false

com.ibm.ws.webcontainer.disallowserveservletsbyclassname:

When the serveServletsByClassnameEnabled property is enabled, it is possible to access servlets directly, resulting in a possible security exposure. Define the following custom property to disallow the use of the serveServletsByClassnameEnabled property across the entire application server level.

Name	Default value
com.ibm.ws.webcontainer.disallowserveservletsbyclassname	false

com.ibm.ws.webcontainer.discernUnavailableServlet: Typically, when the web container receives an UnavailableException, it cannot determine whether the exception was issued from a servlet or a dispatched resource. Therefore, the web container automatically marks the servlet unavailable even if it is the dispatched resource that is unavailable.

If you are running on Version 7.0.0.5 or later, and have set the com.ibm.ws.webcontainer.discernUnavailableServlet custom property to true, any UnavailableException that is issued from a dispatched resource is placed in a wrapper. This wrapper enables the web container to determine whether the exception was issued from the servlet or a dispatched resource. If the exception is not issued by the servlet, the web container does not mark the servlet unavailable.

Name	Default value
com.ibm.ws.webcontainer.discernUnavailableServlet	false

com.ibm.ws.webcontainer.dispatcherRethrowSER: Typically, the RequestDispatcher does not propagate exceptions thrown from dispatched servlets, including JavaServer Pages, back to the servlet doing the dispatching. If you want the RequestDispatcher to throw exceptions back to the servlet doing the dispatching, add the com.ibm.ws.webcontainer.dispatcherRethrowSER custom property to the settings for the web container, and set the property to `true`.

Name	Default value
com.ibm.ws.webcontainer.dispatcherRethrowSER	false

Note: The com.ibm.ws.webcontainer.dispatcherRethrowSError custom property supersedes the com.ibm.ws.webcontainer.dispatcherRethrowSER custom property. When you enable the com.ibm.ws.webcontainer.dispatcherRethrowSError custom property by setting its value to `true`, the com.ibm.ws.webcontainer.dispatcherRethrowSER custom property is also set to true.

com.ibm.ws.webcontainer.dispatcherRethrowSError: When a JavaServer Page (JSP) file contains a compilation error, the runtime error is caught and handled directly by the container. Exceptions are not propagated and addressed by the dispatched JSP resource. With the com.ibm.ws.webcontainer.dispatcherRethrowSError custom property, exceptions are propagated back to the dispatched JSP resource.

Name	Default value
com.ibm.ws.webcontainer.dispatcherRethrowSError	false

Note: The com.ibm.ws.webcontainer.dispatcherRethrowSError custom property supersedes the com.ibm.ws.webcontainer.dispatcherRethrowSER custom property. When you enable the com.ibm.ws.webcontainer.dispatcherRethrowSError custom property by setting its value to `true`, the com.ibm.ws.webcontainer.dispatcherRethrowSER custom property is also set to true.

com.ibm.ws.webcontainer.donotservebyclassname:

The com.ibm.ws.webcontainer.donotservebyclassname custom property specifies a list of classes that cannot be served by the class name.

Name	Default value
com.ibm.ws.webcontainer.donotservebyclassname	none

com.ibm.ws.webcontainer.enabledefaultservletrequestpathelements: To correctly map a request to a default servlet, you must determine the proper servlet path and PathInfo values. The following table shows the affects on the Servlet Path and PathInfo values when you set the `com.ibm.ws.webcontainer.enabledefaultservletrequestpathelements` custom property to a true or false value.

Table 238. Servlet Path and PathInfo values. Values for Servlet Path and PathInfo

Value	Servlet Path value	PathInfo value
true	Set to the contents of the URI after the Context Path	Set to a null value

Table 238. Servlet Path and PathInfo values (continued). Values for Servlet Path and PathInfo

Value	Servlet Path value	PathInfo value
false (Default)	Set to an empty string	Set based on the contents of the URI after the Context Path

com.ibm.ws.webcontainer.enableErrorExceptionTypeFirst: When an exception occurs, the Web container searches for an error page to handle that exception. The default searching order is:

1. Any matching error-code error page
2. Any matching exception-type error page

The matched error-code page is always returned even if there is also a matching exception type error page defined in the web.xml file. To have the Web container search and use the exception-type before the error-code, set this property to `true`.

Name	Default value
com.ibm.ws.webcontainer.enableErrorExceptionTypeFirst	false

com.ibm.ws.webcontainer.enableJspMappingOverride: When a url-pattern is defined in the jsp-property-group of the web.xml, file, it is typically mapped to, and handled by the JavaServer Page (JSP) engine. If you have applications that must override this mapping so that they can handle and serve the JSP content themselves, set the com.ibm.ws.webcontainer.enableJspMappingOverride property to `true`.

Name	Default value
com.ibm.ws.webcontainer.enableJspMappingOverride	false

com.ibm.ws.webcontainer.enableMultiReadOfPostData: Set the com.ibm.ws.webcontainer.enableMultiReadOfPostData custom property to `true` if you want to enable multiple reads of post data. When this property is set to true, the post data can be read multiple times as either an InputStream or Reader, and as parameters.

When the web container is enabled for multiple reads of post data, you can set up an application to complete the following actions if you want that application to re-read post data from the beginning using either an InputStream or Reader:

1. Obtain the InputStream or Reader
2. Read the data
3. Close the InputStrean or Reader

If either the first or third action does not occur, the next read of the post data is not reset to the beginning of that data.

The web container automatically completes this sequence if an application re-reads the post data as parameters.

Name	Default value
com.ibm.ws.webcontainer.enableMultiReadOfPostData	false

com.ibm.ws.webcontainer.extractHostHeaderPort and trusthostheaderport: The getServerPort method relies on the getVirtualPort method of the channel, which returns a port number in the following order:

1. Port number from the request URL
2. Port number from the request host header

This order is compliant with HTTP/1.1 RFC but not with the Java Servlet Specification Version 2.4 API, which requires the port number from the host header to be returned first, if any, or the request URL. The correct returned URL for the above example is: `http://ProxyServer:8888`. The web container was modified to return a port number from the host header, if any, or the URL port that accepted the client connection. You must set the trusthostheaderport and the com.ibm.ws.webcontainer.extractHostHeaderPort custom property to true to return the port number from the request host header first. For example, set these properties in the web.xml file using:

```
trusthostheaderport = true
com.ibm.ws.webcontainer.extractHostHeaderPort = true
```

Or you can set these properties as web container custom properties in the administrative console using the following two sets of name-value pairs:

Name	Default value
com.ibm.ws.webcontainer.extractHostHeaderPort	false
trusthostheaderport	false

com.ibm.ws.webcontainer.finishresponseonclose:

Use the com.ibm.ws.webcontainer.finishresponseonclose custom property to indicate that you want the web container to close a connection when a servlet calls close() on a writer or output stream.

Typically, when a servlet calls close() on a writer or output stream, the web container sends the data that has been written to the writer or output stream to the client, and then waits for the servlet service() method to finish before it closes the connection. This delay might be interpreted as a response completion delay, especially if a servlet service() method does not complete until sometime after the writer or output stream is closed.

Name	Default value
com.ibm.ws.webcontainer.finishresponseonclose	false

com.ibm.ws.webcontainer.ForceDifferentCookiePaths:

Note: When you configure an application to use a cookie to track the session, the default path for the cookie is set to the context root of the application. Therefore, the cookie is only sent to requests that are made to this application. To change the default path to be "/" (forward slash), such that the cookie is sent to requests for any application in this domain, set the ForceDifferentCookiePaths session manager custom property.

Name	Default value
com.ibm.ws.webcontainer.ForceDifferentCookiePaths	false

com.ibm.ws.webcontainer.HTTPOnlyCookies: The com.ibm.ws.webcontainer.HTTPOnlyCookies custom property provides a level of defense against a client-side script accessing a protected cookie and acquiring its content. When you use this custom property, you can prevent Java scripts that run in a browser from accessing all cookies or a particular list of cookies of your choosing. The HTTPOnly attribute is added to each cookie specified in this custom property and enables protection from client-side script access.

gotcha: Specifying com.ibm.ws.webcontainer.HTTPOnlyCookies with no operands means that the HTTPOnly attribute will NOT be added to any cookie, and any client-side Java script running in a browser can access the content of any cookies.

You can specify the following values for this property:

* - An asterisk value means that all cookies are given the HTTPOnly attribute.

A comma delimited list of the specific cookies that are given the HTTPOnly attribute. The HTTPOnly attribute is only given to cookies that are on this list.

The following examples illustrate how to specify these two settings:

```
com.ibm.ws.webcontainer.HTTPOnlyCookies=*
com.ibm.ws.webcontainer.HTTPOnlyCookies=cookieName1,Account3Cookie,JsessionID
```

Note: Cookie names used in specifying com.ibm.ws.webcontainer.HTTPOnlyCookies are case-insensitive.

Name	Default value
com.ibm.ws.webcontainer.HTTPOnlyCookies	none

com.ibm.ws.webcontainer.ignoreinjectionfailure: If a resource or Enterprise JavaBeans (EJB) injection fails during the servlet initialization process, an error message is written to the server log files. However, the error message is not propagated to the client. In addition, the servlet is put into service and it is not reinitialized until its application is restarted. During this time, if a request is received that references the resource, which previously failed to inject, a NullPointerException exception results. Similarly, this problem can occur during the filter and listener initialization processes.

The com.ibm.ws.webcontainer.ignoreinjectionfailure custom property enables you to specify whether to propagate these error messages and whether to put a servlet into service. By default, the custom property is set to `false`, which retains the previously described behavior. To enable the propagation of these injection exceptions to the client and to not put the servlet into service, you must leave this custom property set to `false`.

Name	Default value
com.ibm.ws.webcontainer.ignoreinjectionfailure	false

com.ibm.ws.webcontainer.ignoreInvalidQueryString: When the web container encounters an encoding character in a query string pair that is not valid, it throws an IllegalArgumentException exception and, by default, ignores the entire query string. In applications where every field in the query string is an essential resource, it might not be desirable to ignore the entire query string. If you set the com.ibm.ws.webcontainer.ignoreInvalidQueryString custom property to `true`, the web container ignores query string pairs that are not valid and continues to process valid query string pairs.

Name	Default value
com.ibm.ws.webcontainer.ignoreInvalidQueryString	false

com.ibm.ws.webcontainer.IgnoreSessiononStaticFileRequest: The web container accesses a session for the static file requests involving filters. This action can result in a performance degradation, for example, when running with database session persistence. If you set the com.ibm.ws.webcontainer.IgnoreSessiononStaticFileRequest custom property to `true`, the web container cannot access a session for the static files requests involving filters.

Name	Default value
com.ibm.ws.webcontainer.IgnoreSessiononStaticFileRequest	false

com.ibm.ws.webcontainer.invokeFilterInitAtStartup: The com.ibm.ws.webcontainer.invokeFilterInitAtStartup custom property enables the web container to invoke the init method and initialize a filter during the startup process for an application.

Name	Default value
com.ibm.ws.webcontainer.invokeFilterInitAtStartup	false

com.ibm.ws.webcontainer.invokeFiltersCompatibility:

You might need to use a custom servlet filter with web applications to map files from a one URI to another URI that points to a particular resource. For example, you might map URIs that start with *my_company* to the *my_company*/external directory. Without enabling the *com.ibm.ws.webcontainer.invokeFiltersCompatibility* custom property, the web container does not call any custom servlet filters.

When this custom property is set to true, the web container calls custom servlet filters before looking for welcome files. Also, if the web container cannot find a resource, it calls the custom servlet filters before creating a FileNotFoundException exception. This change enables the web container to verify whether the custom servlet filters modify the path to a resource.

Name	Default value
com.ibm.ws.webcontainer.invokeFiltersCompatibility	false

com.ibm.ws.webcontainer.invokerequestlistenerforfilter: If a web application has defined the listener in the web deployment descriptor xml file, then you must set the com.ibm.ws.webcontainer.invokerequestlistenerforfilter custom property to `true` to have ServletRequestListener invoked when a request is about to enter the filter for that web application.

According to the Java Servlet Specification, a ServletRequestListener should be invoked if a request is about to enter the filter for a web application that has defined the listener in the web deployment descriptor xml file.

Name	Default value
com.ibm.ws.webcontainer.invokerequestlistenerforfilter	false

com.ibm.ws.webcontainer.KeepUnreadPostDataAfterResponseSentToClient: This property indicates whether post data is available to read after the client response is completed, following either the completion of a forward request completes, or a return from a sendRedirect. If this property is set to `true`, post data is available to read after the client response is completed either after a forward request completes, which is the default behavior, or on a return from a sendRedirect,, which occurs when the com.ibm.ws.webcontainer.SendResponseToClientAsPartOfSendRedirect custom property is set to `true`. However, setting this property to `true` requires unread post data to be held in memory until the target resource completes, and increases memory usage.

Name	Default value
com.ibm.ws.webcontainer.KeepUnreadPostDataAfterResponseSentToClient	false

com.ibm.ws.webcontainer.logServletContainerInitializerClassloadingErrors:

Note: When examining the classes of an application to see if they match any of the criteria that is specified by the HandlesTypes annotation of a ServletContainerInitializer, the container might run into class loading problems if one or more of the optional application JAR files are missing. Because the container does not decide whether these types of class loading failures prevent the application from working correctly, it ignores the failures and provides a configuration option that logs them.

Setting this property to true turns on logging.

Name	Default value
com.ibm.ws.webcontainer.logServletContainerInitializerClassloadingErrors	false

com.ibm.ws.webcontainer.mapFiltersToAsterisk:

When processing a request, the web container recognizes servlet mappings to "*" as the same as servlet mappings to "/*". To provide the same behavior with filter mapping, set the com.ibm.ws.webcontainer.mapFiltersToAsterisk custom property to `true`. Setting the com.ibm.ws.webcontainer.mapFiltersToAsterisk custom property to `true` causes the web container to recognize filter mappings to "*" as a filter mapping to "/*". This custom property is not case-sensitive.

Name	Default value
com.ibm.ws.webcontainer.mapFiltersToAsterisk	false

com.ibm.ws.webcontainer.normalizerequesturi: Typically, request URI 404 errors do not occur if a request URI is submitted from a browser, because most modern browsers automatically normalizes a request URI before calling WebSphere Application Server. Therefore, by default, the web container does not normalize a request URI before trying to resolve that URI to an application and servlet mapping.

A request URI, that includes `/./` or `/../` as part of an application context, that has not been normalized, might fail with a 404 error. Similarly, a request URI, that includes `/./`" or `/../` as part of a servlet path, that has not been normalized, fails to match a servlet mapping, which also results in a 404 error, even though the URI is normalized before resolving the URI to a JavaServer Pages (JSP) or static file.

You can set the com.ibm.ws.webcontainer.normalizerequesturi custom property to `true` and the web container normalizes these types of request URIs.

Name	Default value
com.ibm.ws.webcontainer.normalizerequesturi	false

com.ibm.ws.webcontainer.parseUTF8PostData:

If the web container attempts to process a request that includes UTF-8 post data that is not URL encoded, the target resource accesses the post data as parameters. However, the UTF-8 data is not decoded correctly and the result data might be lost.

To resolve this issue, set the com.ibm.ws.webcontainer.parseUTF8PostData custom property to true. When the web container processes parameters, it detects UTF-8 post data that is not URL encoded and includes the data in the parameter values.

To use this function, you must set the value to true.

Name	Default value
com.ibm.ws.webcontainer.parseUTF8PostData	false

com.ibm.ws.webcontainer.provideQStringToWelcomeFile: Typically, when a request is initially sent to the context root of the application, the request is forwarded to a welcome file. If a query string is included in an initial request, it is unavailable to the welcome file if you included the request.getQueryString() attribute in the welcome file. However, the query string is available to the welcome file if you included the javax.servlet.forward.query_string attribute in the welcome file.

If you must use the request.getQueryString() attribute, instead of the javax.servlet.forward.query_string attribute, to make the query string available to the welcome file, add the com.ibm.ws.webcontainer.provideQStringToWelcomeFile custom property to your web container configuration and set the property to `true`.

Name	Default value
com.ibm.ws.webcontainer.provideQStringToWelcomeFile	false

com.ibm.ws.webcontainer.SendResponseToClientAsPartOfSendRedirect: This property indicates whether a response is completed as part of a a sendRedirect request. If this property is set to `true`, a response is completed as part of a sendRedirect request, and any post data associated with the request is not available for a read on return from sendRedirect.

The default value is false.

Name	Default value
com.ibm.ws.webcontainer.SendResponseToClientAsPartOfSendRedirect	false

com.ibm.ws.webcontainer.SendResponseToClientWhenResponseIsComplete: This property indicates whether a response is completed on return from a forward request.

If this property is set to `false`, a response is not completed on return from a forward request. Instead, it is delayed until the target resource completes. Post data is available for a read after the forward completes.

Name	Default value
com.ibm.ws.webcontainer.SendResponseToClientAsPartOfSendRedirect	true

com.ibm.ws.webcontainer.ServeWelcomeFileFromExtendedDocumentRoot: Typically, the first time the web container handles a request for a static welcome page that is not a JavaServer Pages (JSP) file, the web container does not search the ExtendedDocumentRoot for the welcome file unless the request for that welcome file is fully-qualified. If the request is fully-qualified, the web container serves the welcome file, and the context root of the application displays the welcome file. If the request for the static welcome file is not fully-qualified, the web container returns a 404 error, which indicates that the web container did not find the welcome file.

After the web container successfully serves a welcome file, the web container creates a mapping for that welcome file. The web container then uses this mapping to handle future requests for the welcome file, thereby eliminating the need for subsequent requests to be fully-qualified.

If you want the web container to always search an application defined ExtendedDocumentRoot for a welcome file, even if the request is not fully-qualified, you can add the com.ibm.ws.webcontainer.ServeWelcomeFileFromExtendedDocumentRoot custom property to your web container settings, and set this property to `true`.

Name	Default value
com.ibm.ws.webcontainer.ServeWelcomeFileFromExtendedDocumentRoot	false

com.ibm.ws.webcontainer.ServletDestroyWaitTime: By default, when an application is stopped the web container waits up to 60 seconds for each active request for a resource of that application to complete. You can now define the com.ibm.ws.webcontainer.ServletDestroyWaitTime web container custom property to control the amount of time that the web container waits for an active request to complete when the owning application is stopped.

Set the com.ibm.ws.webcontainer.ServletDestroyWaitTime custom property to an integer value, which specifies the number of seconds to wait for a request to complete. The default value is 60 seconds.

Name	Default value
com.ibm.ws.webcontainer.ServletDestroyWaitTime	60

com.ibm.ws.webcontainer.setUnencodedHTMLinsendError: Typically, the web container encodes the specified error messages before formatting them, to prevent Cross-Site Scripting (XSS) attacks on the

client if the application does not sanitize these messages. However the Java Servlet Specification for the sendError(int, String) method, indicates that the server should create the response to look like an HTML-formatted server error page.

If you do not want the web container to encode the specified error messages before formatting them, add the com.ibm.ws.webcontainer.setUnencodedHTMLinsendError custom property to your web container configuration settings, and set the property to `true`.

Name	Default value
com.ibm.ws.webcontainer.setUnencodedHTMLinsendError	false

com.ibm.ws.webcontainer.suppressheadersinrequest: The com.ibm.ws.webcontainer.suppressheadersinrequest custom property can be used to suppress the inclusion of request headers that start with special characters, such as "$" or "_". Some applications do not handle request headers that start with special characters.

The value specified for this custom property is a delimited list of the header prefixes that you want to be suppressed.

Example:

```
com.ibm.ws.webcontainer.suppressheadersinrequest=$WS,_WS
```

Name	Default value
com.ibm.ws.webcontainer.suppressheadersinrequest	none

com.ibm.ws.webcontainer.suppressHtmlRecursiveErrorOutput:

During a recursive error that an application-specified error page cannot handle, the stack trace and error message are output as an HTML page. This information includes class names and program information that the application developer does not want expose to the user.

You can set the com.ibm.ws.webcontainer.suppressHtmlRecursiveErrorOutput web container custom property to suppress the HTML output of the error text, without changing the internal logging of the message. Set the custom property com.ibm.ws. webcontainer.suppressHtmlRecursiveErrorOutput to `true` to disable the HTML output of the error message to the user and present the user with blank page with a 500 error code.

Name	Default value
com.ibm.ws.webcontainer.suppressHtmlRecursiveErrorOutput	false

com.ibm.ws.webcontainer.suppressLastZeroBytePackage: Typically, the last zero byte chunk is used to indicate, to a client, the end of the response data in a chunked encoded transmission. Some applications use this last zero to determine when the response data is completely received, and they can start processing it. If an error occurs in the application after the response headers are sent, the last chunk of data is still sent to the client. The client might not realize that an error has occurred, and attempt to process incomplete data.

If you set the com.ibm.ws.webcontainer.suppressLastZeroBytePackage custom property to `true`, if an error occurs in the application after the response headers are sent, the last chunk of data is not sent to the client.

Name	Default value
com.ibm.ws.webcontainer.suppressLastZeroBytePackage	false

com.ibm.ws.webcontainer.suppressServletExceptionLogging:

If a servlet creates an exception, it is logged to the system console. If you do not want the web container to log servlet- created exceptions, add the com.ibm.ws.webcontainer.suppressServletExceptionLogging custom property to the web container configuration settings, and set the property to `true`.

Name	Default value
com.ibm.ws.webcontainer.suppressServletExceptionLogging	false

com.ibm.ws.webcontainer.throwMissingJspException:

Set the com.ibm.ws.webcontainer.throwMissingJspException custom property to `true` to create a FileNotFoundException when a resource that is included by a JSP file is missing. If this property is not set to `true`, an error page displays.

Name	Default value
com.ibm.ws.webcontainer.throwMissingJspException	false

com.ibm.ws.webcontainer.webgroupvhostnotfound: Error message SRVE0017W states "Web Group not found: {0}", and error message SRVE0255 states "A WebGroup/Virtual Host to handle {0} has not been defined". These messages might be returned when the application that is called to process the request serviced by IBM WebSphere Application Server is not found. You can use the com.ibm.ws.webcontainer.webgroupvhostnotfound custom property to change the text of these messages to text that is more suitable for your environment.

Name	Default value
com.ibm.ws.webcontainer.webgroupvhostnotfound	none

com.ibm.ws.webcontainer.xPoweredBy:

Note: This custom property enables you to configure the value of the X-Powered-By header, which supplies the implementation information of the server.

Name	Default value
com.ibm.ws.webcontainer.xPoweredBy	Servlet/3.0

com.ibm.wsspi.jsp.convertAttrValueToString:

Set the com.ibm.wsspi.jsp.convertAttrValueToString web container custom property to `true` to convert start and end attributes of the repeat tag to strings before they are used.

Name	Default value
com.ibm.wsspi.jsp.convertAttrValueToString	false

com.ibm.wsspi.jsp.disableElCache:

Set the com.ibm.wsspi.jsp.disableElCache web container custom property to `true` to disable the commons-el expression cache if you are experiencing out of memory conditions because the hash maps are held by the expression evaluator.

Name	Default value
com.ibm.wsspi.jsp.disableElCache	false

com.ibm.wsspi.jsp.disableResourceInjection: The resource injection feature accesses resources in applications differently than it did in earlier versions of the product, and causes the compiled method

output to be larger than it was previously. If you have large JSP files that in earlier releases pushed the 65535 byte limit in the translated service method, they might now exceed this limit, causing the compile to fail.

If you encounter this situation, you can either break a large JSP file into smaller JSP files, and use <jsp:include> statements to combine them after they are compiled, or you can add the com.ibm.wsspi.jsp.disableResourceInjection custom property to your web container settings to disable the resource injection function during the JSP translation process. When the com.ibm.wsspi.jsp.disableResourceInjection custom property is set to `true`, the resource injection function is disabled for all applications.

If you only want to disable the resource injection function for specific applications, you can add the disableResourceInjection JSP attribute to the ibm-web-ext.xmi files for those specific applications.

Name	Default value
com.ibm.wsspi.jsp.disableResourceInjection	false

com.ibm.wsspi.jsp.disableTldSearch: The *com.ibm.wsspi.jsp.disableTldSearch* custom property can be used to improve application startup time. By default, when an application starts, the JSP engine searches the application installation directories for the taglib descriptor (TLD) files. This search process might increase the startup time for large applications with many files and directories. To disable this search process, set this property to `true`.

Name	Default value
com.ibm.wsspi.jsp.disableTldSearch	false

com.ibm.wsspi.jsp.enabledoublequotesdecoding:

Set the com.ibm.wsspi.jsp.enabledoublequotesdecoding web container custom property to decode an encoded double quote character if it is embedded in a script function within a JavaServer Pages (JSP) file.

The JSP Container does not decode an encoded double quote character during the translation of a JSP file. Instead, there is a dependency on the browser to decode it. However, when an encoded double quote character exists inside a script function of a tag, the browser cannot decode it. Thus, when this custom property is not set, the encoded double quote character causes the script function to malfunction.

When you set this custom property, the value affects all of your deployed applications. If you want to affect an individual application, set the `enableDoubleQuotesDecoding` JSP attribute to `true` within the `ibm-web-ext.xmi` file in your application.

Name	Default value
com.ibm.wsspi.jsp.enabledoublequotesdecoding	false

DebugSessionCrossover:

The *DebugSessionCrossover* custom property enables code to perform additional checks to verify that only the session associated with the request is accessed or referenced. Messages are logged if any discrepancies are detected.

Note: The use of the *DebugSessionCrossover* property as a web container custom property is deprecated. You can now define it as a session management custom property.

To enable session data crossover detection, set this property to `true`

Name	Default value
DebugSessionCrossover	false

Refer to the HTTP session problems article for additional information.

DecodeUrlAsUTF8:

The UTF-8 encoded URL feature, which provides UTF-8 encoded Uniform Resource Locators (URLs) to support the double-byte characters in URLs is enabled by default. You can prevent the web container from explicitly decoding URLs in UTF-8 and have them use the ISO-8859 standard as per the current HTTP specification by by setting this custom property to `false`.

Name	Default value
DecodeUrlAsUTF8	true

enableInProcessConnections:

Use the enableInProcessConnections custom property to reduce response times and to reduce the number of threads that are used to service a request, which reduces the potential for a deadlock.

There is an optimized communication path between a web services client application and a web container that are located in the same application server process. Requests from the web services client that are normally sent to the web container using a network connection are delivered directly to the web container using an optimized local path. The local path is available because the web services client application and the web container are running in the same process. This optimized communication path is disabled by default. Before enabling this property, make sure that wild cards are not specified for the web container ports. Use specific ports for the web container when the optimized communication path is enabled.

To enable the optimized communication path, set this property to `true`.

Name	Default value
enableInProcessConnections	false

Refer to the Web services client to web container optimized communication topic for additional information.

fileServingEnabled, directoryBrowsingEnabled, and serveServletsByClassnameEnabled:

fileServingEnabled, directoryBrowsingEnabled, and similar properties are global settings for internal servlets. Web application archive (WAR) files that are packaged using third-party tools cannot specify behavior for the services that are exposed by the web container internal servlets.

You can use the fileServingEnabled, directoryBrowsingEnabled, and serveServletsByClassnameEnabled properties to globally enable and disable the fileServing, directoryBrowsing, and serveServletsByClassname functions for internal servlets for all web applications at the web container level.

- Setting the fileServingEnabled, property to `false` disables the fileServing function.
- Setting the directoryBrowsingEnabled, property to `true` enables the directoryBrowsing function.
- Setting the serveServletsByClassnameEnabled property to `true` enables the serveServletsByClassnameEnabled function.

Name	Default value
fileServingEnabled	true
directoryBrowsingEnabled	false

Name	Default value
serveServletsByClassnameEnabled	false

Settings that are defined in an assembly tool take precedence over the global settings that are set through the custom properties at the web container level.

Web application deployment extensions continue to hold configuration information for the services that are provided by the internal servlets, and take precedence over the global settings that are set through the custom properties at the web container level.

ForceSessionIdLengthCheck: Newly-generated session IDs are, by default, 23 characters in length, unless you use the HttpSessionIdLength custom property to specify a different maximum length for your session IDs.

When an incoming request has a session ID that is longer than the expected session ID length, and whose prefix is identical to a pre-existing session ID, the longer ID is used to return a new session. If the length of the session ID on the incoming request is larger then the maximum length specified for your system, such that it exceeds the width of the ID column in the session table column that is used in database persistence, an SQL0302 error occurs.

To prevent the occurrence of these SQL0302 errors, you can add the ForceSessionIdLengthCheck custom property to your web container custom properties and set it to `true`. When this custom property is set to true, the length of a session ID cannot exceed 23 characters. If an incoming request has a session ID that is longer than 23 characters, the first 23 characters are used to return a new session.

Name	Default value
ForceSessionIdLengthCheck	false

ForceSessionInvalidationMultiple:

The *ForceSessionInvalidationMultiple* custom property indicates whether the session manager should wait indefinitely for a request to complete before attempting to invalidate the session, or attempt to invalidate a session after the specified time limit has elapsed.

- If you specify 0 (zero) for this custom property, the session manager waits indefinitely until a request is complete before attempting to invalidate the session.

  If your requests normally are not bound by a response time limit, specify 0 for this property.
- If you specify a positive integer, such as 1, 2, or 3 for this custom property, even if a session is not known to have completed, the session manager attempts to invalidate the session if the indicated time period since the last access occurred has elapsed. This time period is the result of multiplying the value specified for this property and the value specified for the Session Timeout property. For example, if you specify 2 minutes for the Session Timeout property and 2 for the ForceSessionInvalidationMultiple property, the session manager attempts to invalidate the session after 4 minutes.

  If you want to invalidate your sessions after a certain amount of time has elapsed, specify the appropriate positive integer for this property.

Name	Default value
ForceSessionInvalidationMultiple	1

httpsIndicatorHeader:

The custom property *httpsIndicatorHeader* manages HTTPS requests that are forwarded to an application server from an SSL offloader that is used in front of WebSphere Application Server. When an HTTPS request is received by an SSL offloader it is redirected over HTTP to an application server using

WebSphere Application Server. The SSL offloader adds a header indicating the original request was over HTTP. The httpsIndicatorHeader property specifies the request header key name added by the SSL box. The application server checks this indicator to determine if SSL is required. If it determines the request is SSL over HTTP, an HTTPS scheme is chosen.

Name	Default value
httpsIndicatorHeader	none

HttpSessionIdReuse:

The custom property *HttpSessionIdReuse* determines whether the session manager can use the session ID sent from a browser to preserve session data across web applications that are running in an environment that is not configured for session persistence. In a multi-JVM environment, that is not configured for session persistence, setting this property to `true` enables the session manager to use the same session information for all of a user requests, even if the web applications that are handling these requests are governed by different JVM files. The default value for this property is `false`.

Note: The use of the HttpSessionIdReuse property as a web container custom property is deprecated. You should now define this functionality as a session management custom property.

To enable the session manager to use the session ID sent from a browser to preserve session data across web applications that are running in an environment that is not configured for session persistence, set this property to `true`.

Name	Default value
HttpSessionIdReuse	false

listeners:

The servlet specification supports applications registering listeners for servlet-related events on an individual application basis through the `web.xml` descriptor. However, using the listeners custom property, you can enable a server to listen to servlet events across web applications.

To implement global listening, a listener is registered at the web container level and is propagated to all of the installed and new web applications. This global behavior of internal servlet listeners is controlled by the listeners custom property by using the following name-value pair format.

`listeners=`*`listener_class`*

Name	Default value
listeners	none

The value for this property is a string, specifying a comma-separated list of listener classes. The listener that is supplied must implement standard listener classes from the Java Servlet API or IBM listener extension classes.

prependSlashToResource:

WebSphere Application Server 5.x supports Uniform Resource Locators (URLs) without leading front slashes (/) . To preserve compatibility, you can set this custom property to `true`. When this property is set to `true`, the web container ignores the specification and consider URLs without the leading front slash, use the following name-value pair.

Name	Default value
prependSlashToResource	false

trusted: The **trusted** custom property enables the application server to use inbound private headers from the web server plug-in. These inbound private headers notify the application server about the connection to the web server. When you set the custom property to **true**, the application server uses the asserted information on the client certificates. These client certificates are used by the end user to connect to the web server and establish the client information, which is treated as the certificate for the end user. Then, the application server uses the certificate information for authentication purposes when client certificate authentication is used or when the application code accesses the javax.net.ssl.peer_certificates certificates. Because this information is asserted, it is insecure and potentially vulnerable to an attacker that is able to connect directly to the application server and bypass the web server.

Important: If you allow direct connections to the application server and use client certificates, you must set this custom property to **false**.

Name	Default value
trusted	true

Web module deployment settings

Use this page to configure an instance of web module deployment.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Manage Modules** > ***Web_module_instance***.

URI:

Specifies the relative location of the module within the application enterprise archive (EAR) file.

Alternate deployment descriptor:

Specifies the alternate deployment descriptor for the module as defined in the application deployment descriptor according to the Java Platform, Enterprise Edition (Java EE) specification.

Starting weight:

Specifies the order that modules are started. Lower weighted modules are started before higher weighted modules.

If the application deployment descriptor specifies the `<initialize-in-order>true</initialize-in-order>` element, the default starting weights reflect the order that is specified in the deployment descriptor. Otherwise, the defaults are determined based on module type (RAR modules start before EJB modules, which start before web modules).

Class loader order:

Specifies whether the class loader searches in the parent class loader or in the application class loader first to load a class. The standard for development kit class loaders and product class loaders is `Classes loaded with parent class loader first`. By specifying `Classes loaded with application class loader first`, your application can override classes contained in the parent class loader, but this action can potentially result in ClassCastException or LinkageErrors if you have mixed use of overridden classes and non-overridden classes.

The options are `Classes loaded with parent class loader first` and `Classes loaded with local class loader first (parent last)`. The default is to search in the parent class loader before searching in the application class loader to load a class.

Data type String
Default Classes loaded with parent class loader first

Context root for web modules settings

Use this page to specify the context root for web modules during or after installation of an application onto a WebSphere Application Server deployment target.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Context root for web modules**. This page is the same as the Context root for web modules page on the application installation and update wizards.

Web Module:

Specifies the name of a web module in the application that you are installing or that you are viewing after installation.

URI:

Specifies the location of the module relative to the root of the application EAR file.

Context Root:

Specifies the context root of the web application (WAR).

A context root for each web module is defined in the application deployment descriptor during application assembly. Use this field to assign a different context root to a web module. The context root is combined with the defined servlet mapping (from the WAR file) to compose the full URL that users type to access the servlet. For example, if the context root is `/gettingstarted` and the servlet mapping is `MySession`, then the URL is `http://host:port/gettingstarted/MySession`.

Environment entries for web modules settings

Use this page to configure the environment entries of Web modules such as servlets and JavaServer Pages (JSP) files.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Environment entries for web modules**. This page is the same as the Environment entries for web modules page on the application installation and update wizards.

Module:

Specifies the name of a web module.

URI:

Specifies the location of the module relative to the root of the application (EAR file).

Name:

Specifies the name of the environment entry that you are editing or viewing. The environment entry is the `env-entry` property in the web module.

Type:

Specifies a data type for the environment entry defined by the `env-entry` property in the web module.

Description:

Specifies information on the environment entry.

Value:

Specifies an editable value for the environment entry defined by the `env-entry` property in the web module.

The lookup name is displayed in the **Value** column if the lookup name is configured in the application metadata. The lookup name is not editable. If you do not specify a value on this page, the lookup name is used for the value.

Web container troubleshooting tips

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

If you are having problems starting a web module, or accessing resources within a particular web module:

- View the JVM logs and process logs for the application server which hosts the problem web modules, and look for messages in the JVM output file which indicate that the web module has started successfully. You should see messages similar to the following:

```
WebContainer  A SRVE0161I: IBM WebSphere Application Server - Web Container.
Copyright IBM Corp. 1998-2002
WebContainer  A SRVE0169I: Loading Web Module: [module_name]
ApplicationMg A WSVR0221I: Application started: [application_name]
HttpTransport A SRVE0171I: Transport http is listening on port [port_number]
[server_name] open for e-business in profile_root/logs/[server_name]/SystemOut.log
```

- For specific problems that can cause servlets, HTML files, and JavaServer Pages (JSP) files not to be served, refer to the topic, web resource (JSP file, servlet, HTML file, image) does not display .
- For a detailed trace of the run-time behavior of the web container, enable trace for the component com.ibm.ws.webcontainer using `com.ibm.ws.webcontainer*=all`.

If application server related calls fail during Servlet.init method, you can either:

- Initialize the servlet manually by making a single request to that servlet in your browser when the server is ready for e-business instead of starting the servlet upon startup or
- You can choose not to make application server related calls in the servlet's init method.

If the property to start servlets during application server startup is enabled, part of its startup process calls the Servlet.init method on its servlets when you start the web container. Therefore, when the web container is starts and calls the init method, other components such as Naming and Work Load Management may not be fully started yet. As a result, application server related calls may not work since all of the application server components may not be ready yet. Once the application server is 'ready for e-business', it is completely ready.

If none of these steps fixes your problem, check to see if the problem has been identified and documented by looking at the available online support (hints and tips, tech notes, and fixes). If you do not find your problem listed there contact IBM support.

For current information available from IBM Support on known problems and their resolution, refer to the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem. Before opening a PMR, see the IBM Support page.

Disabling servlet pooling: Best practices and considerations

This topic provides usage examples of when you may want to disable servlet pooling. You may want to disable request and response pooling if your application is creating threads inside of the application or if you are concerned about the web container reusing request and response objects.

Disabling request and response pooling

- Application is creating threads inside of the application.

  The Servlet 2.4 specification states the following:

  > **SRV.4.10 Lifetime of the Request Object** Each request object is valid only within the scope of a servlet's service method, or within the scope of a filter's doFilter method. Containers commonly recycle request objects in order to avoid the performance overhead of request object creation. The developer must be aware that maintaining references to request objects outside the scope described above is not recommended as it may have indeterminate results.
  >
  > **SRV.5.6 Lifetime of the Response Object** Each response object is valid only within the scope of a servlet's service method, or within the scope of a filter's doFilter method. Containers commonly recycle response objects in order to avoid the performance overhead of response object creation. The developer must be aware that maintaining references to response objects outside the scope described above may lead to non-deterministic behavior.

- If you are concerned about the web container reuse of reusing request and response objects. Since these objects are reused, there is the potential for two requests in two separate applications to have access to the same request or response object as described in the Thread Safety section of Servlet 2.4.

  > **SRV.2.3.3.3 Thread Safety** Implementations of the request and response objects are not guaranteed to be thread safe. This means that they should only be used within the scope of the request handling thread.
  >
  > References to the request and response objects should not be given to objects executing in other threads as the resulting behavior may be nondeterministic. If the thread created by the application uses the container-managed objects, such as the request or response object, those objects must be accessed only within the servlet's service life cycle and such thread itself should have a life cycle within the life cycle of the servlet's service method because accessing those objects after the service method ends may cause undeterministic problems. Be aware that the request and response objects are not thread safe. If those objects were accessed in the multiple threads, the access should be synchronized or be done through the wrapper to add the thread safety, for instance, synchronizing the call of the methods to access the request attribute, or using a local output stream for the response object within a thread.

It is important to note that disabling pooling prevents the web container from recycling the servlet request and servlet response objects for subsequent requests. This creates additional overhead as a result of an increase in request and response object creation and the subsequent garbage collection of these discarded objects.

JavaServer Pages specific web container custom properties

You can configure name-value pairs of data, where the name is a property key and the value is a string value that you can use to set internal system configuration properties. You can define a new property to configure a setting beyond what is available in the administrative console. This topic includes a list of the available JavaServer Pages custom properties. The JavaServer Pages custom properties are case-sensitive.

You can use the following JSP file-specific web container custom properties:

- "com.ibm.ws.jsp.getparameterreturnemptystring" on page 2553
- "com.ibm.ws.jsp.jdksourcelevel" on page 2553
- "com.ibm.wsspi.jsp.allowjspoutputelementmismatch" on page 2553

- “com.ibm.wsspi.jsp.allowtaglibprefixusebeforedefinition”
- “com.ibm.wsspi.jsp.allowtaglibprefixredefinition” on page 2554
- “com.ibm.wsspi.jsp.allowunmatchedendtag” on page 2554
- “com.ibm.wsspi.jsp.evalquotedandescapedexpression” on page 2554
- “com.ibm.wsspi.jsp.modifyPageContextVariable” on page 2555
- “com.ibm.wsspi.jsp.recompilejsponrestart” on page 2555
- “com.ibm.wsspi.jsp.usecdatatrim” on page 2555
- “com.ibm.wsspi.jsp.usescriptvardupinit” on page 2556
- “com.ibm.wsspi.jsp.usestringcast” on page 2556
- com.ibm.wsspi.jsp.reusepropertygroupconfigoninclude

com.ibm.ws.jsp.getparameterreturnemptystring:

This property is for returning an empty string for actions not set in a JSP file. If a JSP file contains an action and that property has not been set, the JSP engine returns null on WebSphere Application Server Versions 6.0 and 6.1. However, in WebSphere Application Server Version 5.1 an empty string is returned. This custom property was added to provide backwards compatibility. When it is set to true, the value returned on a call to jsp:getProperty is an empty string instead of null for Versions 6.0 and 6.1.

Name com.ibm.ws.jsp.getparameterreturnemptystring

Value true

com.ibm.ws.jsp.jdksourcelevel:

This property is for setting the JDK source level through the administrative console. A JSP engine parameter can be configured for different levels of JDK. However, setting a JSP parameter requires that you set the jdksourcelevel JSP attribute in the web extension file for each web module. However, you can use the com.ibm.ws.jsp.jdksourcelevel custom property to set the JSP attribute globally using the web container custom property. If this attribute is also defined in the web extension file, the property defined in the web extension file supersedes the custom property for that particular application. This custom property is not case-sensitive. The default value is 16.

Name com.ibm.ws.jsp.jdksourcelevel

Value 13, 14, 15, or 16

com.ibm.wsspi.jsp.allowjspoutputelementmismatch:

CTS requirements in previous releases were not applicable to the product, therefore the JSP container supported multiple occurrences of properties in the jsp:output element. In the current release, CTS compliance requires that the JSP container strictly enforces rules about multiple occurrences of properties in the jsp:output element. You can use the com.ibm.wsspi.jsp.allowjspoutputelementmismatch custom property to relax the enforcement of the rule for compatibility with earlier versions.

Name com.ibm.wsspi.jsp.allowjspoutputelementmismatch

Value true

com.ibm.wsspi.jsp.allowtaglibprefixusebeforedefinition:

CTS compliance requires that a tag library directive that defines a prefix must occur before that prefix is used in a custom tag. This rule was not enforced in previous releases because CTS requirements were not required. However, you can use the `com.ibm.wsspi.jsp.allowtaglibprefixusebeforedefinition` custom property to relax the enforcement of the rule for compatibility with earlier versions.

Name com.ibm.wsspi.jsp.allowtaglibprefixusebeforedefinition

Value true

com.ibm.wsspi.jsp.allowtaglibprefixredefinition:

CTS compliance requires that if a tag library prefix is already defined with a different URI within a JSP, the product must create a translation error. This rule was not enforced in previous releases because CTS requirements were not required. However, you can use the com.ibm.wsspi.jsp.allowtaglibprefixredefinition custom property to relax the enforcement of the rule for compatibility with earlier versions.

Name com.ibm.wsspi.jsp.allowtaglibprefixredefinition

Value true

com.ibm.wsspi.jsp.allowunmatchedendtag:

In Version 5 of the product, improper termination of end tags was ignored whereas in Version 6, a translation exception is created. This change of behavior in Version 6 causes problems to users who are migrating their applications, which had improperly terminated end tags, to Version 6. In Version 6, to facilitate migration, a web container property, com.ibm.wsspi.jsp.allowunmatchedendtag, and a JSPAttribute, allowUnmatchedEndTag), are provided. You can enable these properties provides the Version 5 behavior.

Name com.ibm.wsspi.jsp.allowunmatchedendtag

Value true

com.ibm.wsspi.jsp.evalquotedandescapedexpression:

This property is for compiling functions that contain an expression. The JSP translation code was modified to handle escape characters and quotations properly when determining whether to evaluate an expression or to treat it as a literal string. To apply this behavior globally across all web applications, add the following name-value pair as a web container custom property.

Name com.ibm.wsspi.jsp.evalquotedandescapedexpression

Value true

To enable this new behavior for a single application, you must also add the evalquotedandescapedexpression JSP attribute to the `ibm-web-ext.xmi` or `ibm-web-ext.xml` file of the failing application and set the value to `true`.

The following example code shows the attribute in an XMI format:

```
&ltjspAttributes xmi:id="JSPAttribute_1" name="evalquotedandescapedexpression" value="true"/>
```

Note: The attribute ID value must be unique.

The following example code shows the attribute in the `ibm-web-ext.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
 xmlns="http://websphere.ibm.com/xml/ns/javaee"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
 version="1.0">
 <jsp-attribute name="evalquotedandescapedexpression" value="true" />

 <reload-interval value="3"/>
 <auto-encode-requests value="true"/>
 <auto-encode-responses value="true"/>
 <enable-directory-browsing value="true"/>
 <enable-file-serving value="true"/>
 <pre-compile-jsps value="true"/>
```

```
 <enable-reloading value="true"/>
 <enable-serving-servlets-by-class-name value="true" />

</web-ext>
```

Note: Use an assembly tool, such as Rational Application Developer, to modify IBM extension and binding files. You can convert extension and binding files within modules from XMI to XML using the IBM Bindings and Extensions Conversion Tool for Multi-Platforms.

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named ibm-*-ext.xmi or ibm-*-bnd.xmi where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

com.ibm.wsspi.jsp.modifyPageContextVariable:

During the translation phase of a tag file that is compiled, the JSP container implicitly uses the pageContext variable for the PageContext object. The use of the pageContext variable as an implicit variable name in tag files does not comply with the JSP Specification.

If compilation errors occur for applications that use a local pageContext variable in their tag file, set the com.ibm.wsspi.jsp.modifyPageContextVariable custom property to `true` to remove the use of the pageContext variable name in the generated Java code for tag files.

Name com.ibm.wsspi.jsp.modifyPageContextVariable

Value true

com.ibm.wsspi.jsp.recompilejsponrestart:

This property forces JSP files that were compiled at run time to be recompiled every time the application is restarted. This property is helpful if you switch the underlying JSF implementation. This property is best used on development environments.

Name com.ibm.wsspi.jsp.recompilejsponrestart

Value true

com.ibm.wsspi.jsp.usecdatatrim:

You can use the com.ibm.wsspi.jsp.usecdatatrim custom property to trim the text before creating CDATA section which eliminates the extra white spaces addedUsers Affected: WebSphere Application Server Version 6.0 users who are migrating their applications from Version 5 to Version 6 and whose JSPs use nesting tags in separate lines. Problem Description: JSP files that use nesting tags have extra lines in the code generated for this section. In Version 6, you could eliminate the extra spaces by appending all the lines into one using: '' The extra line is added in the Java code generated because the text is not trimmed

before creating the CDATA section. Therefore, a switch is provided to enable the trimming of the text before creating the CDATA section. You can set the JSP attribute, useCDataTrim, at the web module level or set the com.ibm.wsspi.jsp.usecdatatrim web container property at the web container level using the following name-value pair.

Name com.ibm.wsspi.jsp.usecdatatrim

com.ibm.wsspi.jsp.usescriptvardupinit:

The code generated for a JSP file assumed that the same tag variables to be declared two or more times in an If-Else condition, even if the variable had a page scope. The com.ibm.wsspi.jsp.usescriptvardupinit custom property enables this feature for all the applications deployed on a particular server. If the compatibility feature is required only for a specific application, enable the useScriptVarDupInit JSP attribute. If both the options are set, then the JSP attribute takes preference over the web container custom property.

Name com.ibm.wsspi.jsp.usescriptvardupinit

Value true

com.ibm.wsspi.jsp.usestringcast:

The com.ibm.wsspi.jsp.usestringcast property explicitly adds a String cast to the relative path before inclusion when you migrate Version 5.1 applications. In Version 6.0.x, the generated Java source for the JSP file did not add the "implicit" cast, for return types of type String when the request.getAttribute method is called. When a JSP file includes a resource whose relative path does not evaluate to a String, then the include fails as the include takes only a String as a relative path of the resource. You can set the com.ibm.wsspi.jsp.usestringcast custom property, which would affect all the deployed applications or you can set the property as a JSPAttribute, useStringCast, in the extensions file, which affects only the application for which the property is set.

Name com.ibm.wsspi.jsp.usestringcast

Value true

com.ibm.wsspi.jsp.reusepropertygroupconfigoninclude:

Note: Most properties that are defined in a JSP property group apply to an entire translation unit, for example, the requested JSP file that is matched by its URL pattern and all the files it includes using the include directive. The exceptions are the page-encoding and is-xml properties, which apply separately to each JSP file that is matched by its URL pattern. To revert the behavior to a setting before WebSphere Application Server Version 8.0, set the custom property to `true` to apply the two property values to the entire translation unit.

Name com.ibm.wsspi.jsp.reusepropertygroupconfigoninclude

Value false

Configuring JSP engine parameters

Learn about how to add, change or delete JSP engine configuration parameters.

About this task

The following note applies to the file references with a .xmi extension in this topic:

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or

module. An IBM extension or binding file is named ibm-*-ext.xmi or ibm-*-bnd.xmi where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

WebSphere Application Server does not support the modification of deployment descriptor extension parameters through the Administrative Console or through administrative scripting.

Note: Use an assembly tool, such as Rational Application Developer, to modify IBM extension and binding files. You can convert extension and binding files within modules from XMI to XML using the IBM Bindings and Extensions Conversion Tool for Multi-Platforms.

To add, change or delete JSP engine configuration parameters, complete the following steps:

Procedure

1. Open the `WEB-INF/ibm-web-ext.xmi` or `WEB-INF/ibm-web-ext.xml` file.

   JSP engine configuration parameters are stored in a web module's configuration directory or in a web modules's binaries directory in the `WEB-INF/ibm-web-ext.xmi` or `WEB-INF/ibm-web-ext.xml` file. Open the `WEB-INF/ibm-web-ext.xmi` or `WEB-INF/ibm-web-ext.xml` file from:

   - The configuration directory, as in the following examples:

     profile_root/config/cells/*cellName*/applications/*enterpriseAppName*/deployments/*deployedName*/*webModuleName*/WEB-INF/ibm-web-ext.xmi

     profile_root/config/cells/*cellName*/applications/*enterpriseAppName*/deployments/*deployedName*/*webModuleName*/WEB-INF/ibm-web-ext.xml
   - The binaries directory if an application was deployed into WebSphere Application Server with the flag "Use Binary Configuration" set to true. Example of a binaries directory is:

     profile_root/installedApps/*nodeName*/*applicationName*.ear/applicationName.war/WEB-INF/ibm-web-ext.xmi

     profile_root/installedApps/*nodeName*/*applicationName*.ear/applicationName.war/WEB-INF/ibm-web-ext.xml
2. Edit the `WEB-INF/ibm-web-ext.xmi` or `WEB-INF/ibm-web-ext.xml` file.
   - To add configuration parameters to the `WEB-INF/ibm-web-ext.xmi` file, use the following format:

     ```
     xmi:id="JSPAttribute_6" name="parametername" value="parametervalue"/>
     ```
   - To add configuration parameters to the `WEB-INF/ibm-web-ext.xml` file, use the following format:

     ```
     <jsp-attribute name="parametername" value="parametervalue"/>
     ```
   - To delete configuration parameters, either delete the line from the file, or enclose the statement with `<!-- -->` tags.
3. Save the file.
4. Restart the Enterprise Application. It is not necessary to restart the server for parameter changes to take effect. However, some JSP engine configuration parameters affect the Java source code that is generated for a JSP. If such a parameter is changed, then you must retranslate the JSP files in the web module to regenerate Java source. You can use the batch compiler to retranslate all JSP files in a web module. The batch compiler uses the JSP engine configuration parameters that you have set in the `ibm-web-ext.xmi` or `ibm-web-ext.xml` file, unless you specifically override them. The topic, JSP engine configuration parameters, identifies the parameters that affect the generated Java source.

Example

The following is a sample of the WEB-INF/ibm-web-ext.xmi file. The lines in bold text are JSP engine configuration parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<webappext:WebAppExtension xmi:version="2.0" xmlns:xmi=http://www.omg.org/XMI
   xmlns:webappext="webappext.xmi" xmlns:webapplication="webapplication.xmi" xmi:id="WebAppExtension_1"
   reloadInterval="9" reloadingEnabled="true" defaultErrorPage="error.jsp" additionalClassPath=""
   fileServingEnabled="true" directoryBrowsingEnabled="false" serveServletsByClassnameEnabled="true"
   autoRequestEncoding="true" autoResponseEncoding="false"
   <webApp href="WEB-INF/web.xml#WebApp_1"/>
   <jspAttributes xmi:id="JSPAttribute_1" name="useThreadTagPool" value="true"/>
   <jspAttributes xmi:id="JSPAttribute_2" name="verbose" value="false"/>
   <jspAttributes xmi:id="JSPAttribute_3" name="deprecation" value="false"/>
   <jspAttributes xmi:id="JSPAttribute_4" name="reloadEnabled" value="true"/>
   <jspAttributes xmi:id="JSPAttribute_5" name="reloadInterval" value="5"/>
   <jspAttributes xmi:id="JSPAttribute_6" name="keepgenerated" value="true"/>
   <!--<jspAttributes xmi:id="JSPAttribute_7" name="trackDependencies" value="true"/> -->

</webappext:WebAppExtension>
```

The following is a sample of the WEB-INF/ibm-web-ext.xml file. The lines in bold text are JSP engine configuration parameters.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">

   <default-error-page uri="error.jsp"/>
   <jsp-attribute name="useThreadTagPool" value="true" />
   <jsp-attribute name="verbose" value="false" />
   <jsp-attribute name="deprecation" value="false" />
   <jsp-attribute name="reloadEnabled" value="true" />
   <jsp-attribute name="reloadInterval" value="5" />
   <jsp-attribute name="keepgenerated"  value="true" />
   <jsp-attribute name="trackDependencies" value="true" />
   <reload-interval value="9"/>
   <auto-encode-requests value="true"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="false"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="true"/>
</web-ext>
```

Attention: The integer **n** in **JSPattribute_n** has to be unique within the file.

JSP engine

The WebSphere Application Server JavaServer Pages (JSP) engine is the implementation of the JavaServer Pages Specification.

WebSphere Application Server Version 8.0 supports the JSP 2.1 specification.

The JSP engine

- Validates JSP source, both classic and XML styles
- Translates JSP source to Java classes
- Compiles Java classes, reporting any errors
- Generates Java classes for any tag files that are used by the JSP
- Interfaces with the web container to load JSP class files

- Supports JSP batch compilation, JSP compilation during application installation, and JSP compilation during the build process of customer applications, through an Ant task.
- Loads class files, and manage life-cycle (reloading, unloading as necessary)
- Supports debugging of JavaServer Pages files through support for JSR 45 (Debugging Support for Other Languages)

JSP engine configuration parameters

In WebSphere Application Server, you can configure the JavaServer Pages (JSP) engine configuration parameters for optimal performance in a production server environment and for the needs of developers in a development environment.

The JSP engine parameters are case sensitive. If the value specified for a parameter is comprised of two or more words separated by spaces, you must add quotation marks around the value. Some parameters affect the Java source that is generated for a JSP or tag file. These parameters are identified by the statement "This parameter requires regeneration of Java source." This statement indicates that if the configuration parameter is modified, the new value for the parameter does not have any effect until the JSP files are retranslated and the Java sources are recompiled.

Note: Use an assembly tool, such as Rational Application Developer, to modify IBM extension and binding files. You can convert extension and binding files within modules from XMI to XML using the IBM Bindings and Extensions Conversion Tool for Multi-Platforms.

compileWithAssert:

Specifies whether the generated Java classes should contain support for the Developer Kit, Java Technology Edition 1.4 Assertion facility. The effect of setting this parameter to true is that the –source 1.4 option is passed to the Java compiler. The default for this parameter is `false`. This parameter requires regeneration of Java source.

classdebuginfo:

Indicates whether the compiler includes debugging information in the generated class file. When you set this parameter to true, the –g option is passed to the Java compiler. The default for this parameter is `false`. This parameter requires regeneration of Java source.

convertAttrValueToString:

Specifies whether to convert start and end attributes of the repeat tag to strings before they are used. The default for this parameter is `false`. This parameter requires regeneration of Java source.

deprecation:

Specifies whether the compiler generates deprecation warnings when compiling the generated Java source. When you set this parameter to true, the -deprecation option is passed to the Java compiler. The default for this parameter is `false`. This parameter requires regeneration of Java source.

disableElCache:

Set the com.ibm.wsspi.jsp.disableElCache web container custom property to `true` to disable the commons-el expression cache if you are experiencing out of memory conditions because the hash maps are held by the expression evaluator. The default for this parameter is `false`.

disableJspRuntimeCompilation:

If this option is set to true, the JSP engine at runtime does not translate and compile JSP files; the JSP engine loads only precompiled class files. JSP source files do not need to be present in order to load

class files. When this option is set to true, you can install an application without JSP source, but the application must have precompiled class files. There is a web container custom property with the same name that is used to determine the behavior of all web modules installed in a server. If both the web container custom property and the JSP engine option are set, the JSP engine option takes precedence. The default for this parameter is `false`.

disableTldSearch:

Set this option is set to `true` to prevent the JSP engine from searching the application installation directories for the taglib descriptor (TLD) files when an application starts. If this option is set to false, when an application starts, the JSP engine searches the application installation directories for the TLD files. The default for this parameter is `false`.

There is a web container custom property, com.ibm.wsspi.jsp.disableTldSearch, that can be used to specify whether, when an application starts, the JSP engines in all web modules installed in a server do not search the application installation directories for the TLD files. If conflicting values are set for the web container custom property and the JSP engine option, the setting for the JSP engine option takes precedence.

Note: This option is only available in version levels 7.0.0.3, and higher.

evalQuotedAndEscapedExpression:

Set this option to true to handle escape characters and quotations properly when determining whether to evaluate an expression.

During the translation phase of a JSP compile, expressions are evaluated by the JSP engine. Characters, such as the escape character (\) or nested quotations, single or double, causes the JSP file translation to fail. For example, when you use functions that contain an expression such as

```
<input type="text" value="${fn:substring('1234567', 0,4)}"/>
```

Because of the double quote directly before the fn:substring statement, the JSP file fails to compile because the translator did not add the function mapper to the generated Java class. Also, if a dollar sign ($) was escaped using the backslash (\$), the translator still attempts to evaluate the expression instead of treating it as a literal string. To handle escape characters and quotations properly, you must set evalQuotedAndEscapedExpression to `true` in the `ibm-web-ext.xmi` or `ibm-web-ext.xml` file of the failing application.

The following code sample shows an entry in the `ibm-web-ext.xmi` file:

```
<jspAttributes xmi:id="JSPAttribute_1"
name="evalQuotedAndEscapedExpression" value="true"/>
```

The following code sample shows an entry in the `ibm-web-ext.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">
   <file-serving-attribute name="extendedDocumentRoot" value="true" />
   <jsp-attribute name="evalQuotedAndEscapedExpression" value="true" />
   <jsp-attribute name="extendedDocumentRoot" value="/opt/extDocRootDir" />
   <jsp-attribute name="extendedDocumentRoot" value="${MY_CUSTOM_VARIABLE}" />

   <reload-interval value="3"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="true"/>
```

```
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

To apply this behavior globally across all web applications, you can set the com.ibm.wsspi.jsp.evalQuotedAndEscapedExpression web container custom property to true.

extendedDocumentRoot:

Use the extended document root facility when applications require access to files outside of the application web application archive (WAR) directory. This facility enables you to configure an application with one or more directory paths from which you can serve static files and JSP files. You can use this attribute when an application requires access to files that exist outside of the web application archive (WAR) directory. For example, if several applications require access to a set of common files, you can place the common files in a directory to which you can link each application as an extended document root directory.

To configure an application with an extended document root, add an extendedDocumentRoot attribute as a file-serving attribute to the `ibm-web-ext.xmi` or `ibm-web-ext.xml` file for the application. The value of this attribute is a comma-delimited list of directories that serve as the root directory location for the static files.

The following entry is an example within the `ibm-web-ext.xmi` file:

```
<fileServingAttributes xmi:id="FileServingAttribute_1" name="extendedDocumentRoot" value="/opt/extDocRootDir"/>
```

The following examples are based on the previous entry in the `ibm-web-ext.xmi` file with the attribute set to the `/opt/extDocRootDir` value:

- A request for the http://localhost:9080/*context_root*/sample.html resource requires that the `sample.html` file is located in the `/opt/extDocRootDir/sample.html` directory structure.
- A request for the http://localhost:9080/*context_root*/myDir/sample.gif resource requires that the `sample.gif` file is located in the `/opt/extDocRootDir/myDir/sample.gif` directory structure.

The following entry is an example within the `ibm-web-ext.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">
   <file-serving-attribute name="extendedDocumentRoot" value="/opt/extDocRootDir" />
   <jsp-attribute name="evalQuotedAndEscapedExpression" value="true" />
   <jsp-attribute name="extendedDocumentRoot" value="/opt/extDocRootDir" />
   <jsp-attribute name="extendedDocumentRoot" value="${MY_CUSTOM_VARIABLE}" />
   <reload-interval value="3"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="true"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

Important: To serve static files from an extended document root directory, you must enable file serving.

To configure an application with an extended document root from which JSP files are served, add an extendedDocumentRoot attribute as a JSP attribute to the `ibm-web-ext.xmi` or `ibm-web-ext.xml` file. The value of this attribute is a comma-delimited list of directories that serve as the root directory location for the JSP files.

The following entry is an example within the `ibm-web-ext.xmi` file:

```
<jspAttributes xmi:id="JSPAttribute_1" name="extendedDocumentRoot" value="/opt/extDocRootDir"/>
```

The following example shows the entry within the `ibm-web-ext.xml` file:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">
   <file-serving-attribute name="extendedDocumentRoot" value="/opt/extDocRootDir" />
   <jsp-attribute name="evalQuotedAndEscapedExpression" value="true" />
   <jsp-attribute name="extendedDocumentRoot" value="/opt/extDocRootDir" />
   <jsp-attribute name="extendedDocumentRoot" value="${MY_CUSTOM_VARIABLE}" />
   <reload-interval value="3"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="true"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

You can also use an extendedDocumentRoot attribute to define a WebSphere variable on multiple nodes to the appropriate directory.

`ibm-web-ext.xmi` example:

```
<jspAttributes xmi:id="JSPAttribute_2" name="extendedDocumentRoot"
    value="${MY_CUSTOM_VARIABLE}"/>
```

`ibm-web-ext.xml` example:

```
 <jsp-attribute name="extendedDocumentRoot"
    value="${MY_CUSTOM_VARIABLE}" />
```

where *MY_CUSTOM_VARIABLE* is the WebSphere variable that you want to define on multiple nodes.

The following example shows a `ibm-web-ext.xmi` file that defines an extended document root both as a file-serving attribute and a JSP attribute:

```
<?xml version="1.0" encoding="UTF-8"?>
<com.ibm.ejs.models.base.extensions.webappext:WebAppExtension xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:com.ibm.ejs.models.base.extensions.webappext="webappext.xmi" xmi:id="WebAppExtension_1"
          reloadInterval="3"
          reloadingEnabled="true"
          fileServingEnabled="true">
      <webApp href="WEB-INF/web.xml#WebApp_ID"/>
      <fileServingAttributes xmi:id="FileServingAttribute_1" name="extendedDocumentRoot"
       value="/opt/extDocRootDir"/>
      <jspAttributes xmi:id="JSPAttribute_1" name="extendedDocumentRoot"
       value="/opt/extDocRootDir"/>
<com.ibm.ejs.models.base.extensions.webappext:WebAppExtension>
```

The following example shows a `ibm-web-ext.xml` file that defines an extended document root both as a file-serving attribute and a JSP attribute:

```
<?xml version="1.0" encoding="UTF-8"?>
<web-ext
   xmlns="http://websphere.ibm.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://websphere.ibm.com/xml/ns/javaee http://websphere.ibm.com/xml/ns/javaee/ibm-web-ext_1_0.xsd"
      version="1.0">
   <file-serving-attribute name="extendedDocumentRoot" value="/opt/extDocRootDir" />
   <jsp-attribute name="evalQuotedAndEscapedExpression" value="true" />
   <jsp-attribute name="extendedDocumentRoot" value="/opt/extDocRootDir" />
   <jsp-attribute name="extendedDocumentRoot" value="${MY_CUSTOM_VARIABLE}" />
```

```
   <reload-interval value="3"/>
   <auto-encode-requests value="false"/>
   <auto-encode-responses value="false"/>
   <enable-directory-browsing value="false"/>
   <enable-file-serving value="true"/>
   <pre-compile-jsps value="false"/>
   <enable-reloading value="true"/>
   <enable-serving-servlets-by-class-name value="false" />
</web-ext>
```

If the request is a valid partial request for a welcome file, a 404 error is returned. If the JSP file is located inside a JAR file and the reloadEnabled attribute value is true, the time stamp of the JAR file is used for isOutDated checks for recompile purposes. The default for this parameter is null.

Note: For IBM extension and binding files, the .xmi or .xml file name extension is different depending on whether you are using a pre-Java EE 5 application or module or a Java EE 5 or later application or module. An IBM extension or binding file is named ibm-*-ext.xmi or ibm-*-bnd.xmi where * is the type of extension or binding file such as app, application, ejb-jar, or web. The following conditions apply:

- For an application or module that uses a Java EE version prior to version 5, the file extension must be .xmi.
- For an application or module that uses Java EE 5 or later, the file extension must be .xml. If .xmi files are included with the application or module, the product ignores the .xmi files.

However, a Java EE 5 or later module can exist within an application that includes pre-Java EE 5 files and uses the .xmi file name extension.

The `ibm-webservices-ext.xmi`, `ibm-webservices-bnd.xmi`, `ibm-webservicesclient-bnd.xmi`, `ibm-webservicesclient-ext.xmi`, and `ibm-portlet-ext.xmi` files continue to use the .xmi file extensions.

ieClassId:

Indicates the Java plug-in COM class ID for Internet Explorer.

The `<jsp:plugin>` tags use this value. The default classid is `clsid:8AD9C840-044E-11D1-B3E9-00805F499D93`

javaEncoding:

Specifies the encoding that is used when the `.java` file is generated, and when it is compiled by the Java compiler. Set this parameter when the page encoding of your JSP pages is not UTF-8 compatible. When javaEncoding is set, the encoding is passed to the Java compiler through the -encoding argument. Note that encoding is not supported by Jikes. The default is UTF-8. This parameter requires regeneration of Java source.

jdkSourceLevel:

This JSP engine parameter was introduced in WebSphere Application Server version 6.1 to support JDK 5. Use this parameter instead of the compileWithAssert parameter, although compile WithAssert still works in version 6.1.

The default value for this parameter is 16. This parameter requires regeneration of Java source. The following are jdkSourceLevel paramater values:

- **13** - This value will disable all new language features of JDK 1.4, JDK 5.0 and JDK 6.0.

- **14** - This value will enable the use of the assertion facility and will disable all new language features of JDK 5.0 and JDK 6.0.
- **15** - This value will enable the use of the assertion facility and will disable all new language features of JDK 6.0.
- **16** - This value will enable the use of the new features of JDK 6.0.

jsp.file.extensions:

For JSP files with extensions other than the four standard extensions, *.jsp, *.jspx, *.jsw, and *.jsv, you can configure this the extensions using this parameter. These extensions are added to the standard extensions.

The preferred method for doing this is to create a `<jsp-property-group>` in web.xml, and add a `<url-pattern>` tag for each extension.

The JSP engine can handle a list of file extensions that is separated by a colon or semi-colon. For example, *.ext1;*.ext2:*.extn

keepgenerated:

Indicates that the Java files generated by the JSP compiler during the translation phase of the processing are retained. The default for this parameter is `false`. This parameter requires regeneration of Java source.

keepGeneratedclassfiles:

Indicates that the class files generated by the JSP compiler during the translation phase of the processing are retained. The default for this parameter is `true`. This parameter requires regeneration of Java source.

modifyPageContextVariable:

During the translation phase of a tag file that is compiled, the JSP container implicitly uses the pageContext variable for the PageContext object. The use of the pageContext variable as an implicit variable name in tag files does not comply with the JSP Specification. If compilation errors occur for applications that use a local pageContext variable in their tag file, set the modifyPageContextVariable attribute to`true` to remove the use of the pageContext variable name in the generated Java code for tag files.

recompileJspOnRestart:

Determines whether a JSP file is retranslated and recompiled after application startup for the first time the file is requested. If recompileJspOnRestart is false, a JSP file is still compiled, if necessary, on the first request to that JSP file unless the parameter disableJspRuntimeCompilation is true. The default for this parameter is false.

reloadEnabled:

Determines whether or not a JSP file is translated and compiled at runtime if the JSP file or its dependencies (see trackDependencies) are modified.

If reloadEnabled is false, a JSP file is still compiled, if necessary, on the first request to it unless the parameter disableJspRuntimeCompilation is true. The default for this parameter is `false`.

If this JSP engine parameter is not specified, the equivalent web container parameter for web module class reloading is used. However, for an application whose deployment descriptor is at the Servlet 2.2 level, the default is true. This is done for the support of applications being migrated from WebSphere Application Server Version 4.x.

reloadInterval:

If reloading is enabled, reloadInterval determines the delay between checks to see if a JSP file is outdated.

For example, if reloadInterval is 5, the JSP engine checks to see if a JSP file is outdated only when the last such check was done more than 5 seconds prior to the current request for the JSP file. The larger the reloadInterval, the less frequently the JSP engine checks for the need to reload a JSP file. If this JSP engine parameter is not specified, the equivalent web container parameter for web module class reloading is used. However, for an application whose deployment descriptor is at the Servlet 2.2 level, the default is 5 seconds. This is done for the support of applications being migrated from WebSphere Application Server Version 4.x.

reusePropertyGroupConfigOnInclude:

Note: Most properties that are defined in a JSP attribute group apply to an entire translation unit, for example, the requested JSP file that is matched by its URL pattern and all the files it includes using the include directive. The exceptions are the page-encoding and is-xml properties, which apply separately to each JSP file that is matched by its URL pattern. To revert the behavior to a setting prior to WebSphere Application Server Version 8.0, set the attribute to `true` to apply the two property values to the entire translation unit.

scratchdir:

Specifies the directory where the generated class files are created.

The system property com.ibm.websphere.servlet.temp.dir is used to set the scratchdir option on a server-wide basis. The JSP engine scratchdir parameter takes precedence over the system property com.ibm.websphere.servlet.temp.dir. The default for this parameter is *profile_root*/`temp`. This parameter requires regeneration of Java source.

gotcha: Do not specify a directory path that places the JSP temporary directory at the same level or underneath the configuration temporary directory. The default location for the configuration temporary directory is *profile_home*/config/temp.

For example, if you change the location of the JSP temporary directory to also be *profile_home*/config/temp. or if you place the JSP temporary directory underneath the configuration temporary directory at *profile_home*/config/temp/temp, processing errors occur.

trackDependencies:

If reloading is enabled, trackDependencies determines whether the JSP engine tracks modifications to the requested JavaServer Pages files dependencies as well as to the JSP file itself.

The dependencies tracked by the JSP engine are :

1. files statically included in the JSP file
2. tag files referenced in the JSP file (excluding tag files that are in JARs)
3. TLD files referenced in the JSP file (excluding TLDs that are in JARs)

The default is false.

useFullPackageNames:

If useFullPackageNames is true, the JSP engine generates and loads JSP classes using full package names.

The default is to generate all JSP classes in the same package. (For more information, refer to the Packages and directories for generated .java and .class files topic). The JSP engine's class loader knows how to load JSP classes when they are all in the same package.

The default method of generating all JSP classes in the same package has the benefit of generating smaller file-system paths. Full package names has the benefit of enabling the configuration of precompiled JSP class files as servlets in the `web.xml` file without the use of the jsp-file attribute, resulting in a single class loader, the web application's class loader, that is used to load all such JSP classes. Similarly, when the JSP engine's configuration attributes useFullPackageNames and disableJspRuntimeCompilation are both true, a single class loader is used to load all JSP classes, even if the JSP files are not configured as servlets in the `web.xml` file.

When useFullPackageNames is set to true, the batch compiler generates a `generated_web.xml` file in the web module's `WEB-INF` directory. This file contains servlet configuration information for each JSP file that was successfully translated and compiled. The information can optionally be copied into the web module `web.xml` file so that the JSP files are loaded as servlets by the web container. Note that if a JSP file is configured as a servlet in this way, no reloading of the JSP file is done at runtime if the JSP file is modified. This is because the JSP file is treated as a regular servlet and requests for it do not pass through the JSP engine. This parameter requires regeneration of Java source.

useImplicitTagLibs:

The JSP engine implicitly recognizes tsx and jsx as tag library prefixes for tag libraries supplied by the JSP engine. If tsx or jsx are used as prefixes for a customer's tag library, the customer's tag library overrides the implicit tag library. However, the implicit tag library is still cached by the JSP engine. Explicitly setting this parameter to false tells the engine not to cache the implicit tag library, and save resources. The default for this parameter is true.

The default URL for the tsx tags is http://websphere.ibm.com/tags/tsx. The default URI for the jsx tags is http://websphere.ibm.com/tags/jsx.

You might need to define the tsx or jsx tag library prefix with a different URI than its default URI. Also, you might need to define the same library prefix with a different URI. In these two situations, you can set the com.ibm.wsspi.jsp.allowtaglibprefixredefinition custom property to avoid translation errors. For more information, see the documentation about JavaServer Pages custom properties.

useInMemory:

Specifies that the JSP engine translate and compile Java code in the system memory.

When this option is not set, the JSP engine must perform the following steps:

1. Write the translated Java file to the file system
2. Load the Java file from the file system
3. Compile the code into a class file
4. Write the class to the file system
5. Load the class file into a classloader.

Note: No .class file or .java file will be written to the system disk. For debugging or creating a JAR file from precompiled JSP code, you will need to disable this option.

useJikes:

Specifies whether Jikes is used for compiling Java sources.

NOTE: Jikes is not shipped with WebSphere Application Server. The default for this parameter is false. This parameter requires regeneration of Java source.

usePageTagPool:

*Enables or disables the reuse of custom tag handlers on an individual JavaServer Pages basis. The default for this parameter is false. This parameter requires regeneration of Java source.

useThreadTagPool:

When thread-level tag handler pooling is used, tag handlers may be reused among separate occurrences of a custom action across all JSP pages in a single web module across separate requests. The default for this parameter is `false`. This parameter requires regeneration of Java source.

Enabling custom tag handler reuse might reveal problems in the tag handler code with regard to the tag's ability to be reused. A custom tag handler should always do two things:

- The release method of the tag handler should reset its state and release any private resources that it might have used. The JSP engine ensures the release method is called before the tag handler is garbage collected.
- In the doEndTag method, all instance states associated with this instance must be reset.

verbose:

Indicates that the compiler generates verbose output when compiling the generated Java source code. The effect of setting this parameter to true is that the -verbose option is passed to the Java compiler. The default for this parameter is false. This parameter requires regeneration of Java source.

*Enabling custom tag handler reuse might reveal problems in the tag handler code with regard to the tag's ability to be reused. A custom tag handler should always do two things:

- The release method of the tag handler should reset its state and release any private resources that it might have used. The JSP engine ensures the release method is called before the tag handler is garbage collected.
- In the doEndTag method, all instance states associated with this instance must be reset.

  CAUTION:
  When using page or thread tag pooling, the doEndTag method is not called in the case of an exception, and if there is service state that must be cleared, then the TryCatchFinally interface should be implemented.

JavaServer Pages troubleshooting tips

Use this tips to troubleshoot problems with JavaServer Pages.

JavaServer Pages source code shown by the web server

If you share the document root of the WebSphere Application Server with the web server document root, a security exposure can result as the web server might display the JavaServer Pages (JSP) source file as plain text.

Problem

You can use the WebSphere Web server plug-in set of rules to determine whether a given request will be handled by the WebSphere Application Server. When an incoming request fails to match those rules, the web server plug-in returns control to the web server so that the web server can fulfill the request. In this case, the unknown host header causes the web server plug-in to return control to the web server because the rules do not indicate that the WebSphere Application Server should handle it. Therefore, the web server looks for the request in the web server document root. Since the JSP source file is stored in the document root of the web server, the web server finds the file and displays it as plain text.

Suggested solution

Move the WebSphere Application Server JSP source file outside of the web server document root. Then, when this request comes in with the unknown host header, the plug-in returns control to the web server and the JSP source file is not found in the document root. Therefore, the web server returns a `404 File Not Found` error rather than the JSP source file.

Problems displaying double-byte character set (DBCS) characters when using the @include directive

JavaServer Pages files that use the @include directive might experience problems when displaying double-byte character set (DBCS) characters. Some applications that are migrated to WebSphere Application Server Version 6.0 and above might need to be modified to comply with the JSP 2.0 specification as a result of backwards compatibility issues. The JSP 2.0 specification requires that each statically included resource must set a page encoding or content type because the character encoding for each file is determined separately, even if one file includes another using the include directive.

Problems using the JavaServer Pages (JSP) engine

If you are having difficulty using the JavaServer Pages (JSP) engine, try these steps:

1. Determine whether other resources such as `.html` files or servlets are being requested and displayed correctly. If they are not, the problem probably lies at a deeper level, such as with the HTTP server.
2. If other resources are being displayed correctly, determine whether the JSP processor has started normally:
   - Browse the JVM logs of the server hosting the JSP files you are trying to access. The following messages indicate that the JSP processor has started normally:

     ```
     Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
     was initialized successfully.
     Extension Processor [class com.ibm.ws.jsp.webcontainerext.JSPExtensionProcessor]
     has been associated with patterns [*.jsp *.jspx *.jsw *.jsv ].
     ```

     If the JSP processor fails to load, you will see a message such as

     ```
     No Extension Processor found for handling JSPs.
     JSP Processor not defined. Skipping : jspfilename.
     ```

     in the *root_dir*/logs/*server_name*/SystemOut.log file
3. If the JSP engine has started normally, the problem may be with the JSP file itself.
   - The JSP may have invalid JSP syntax and could not be processed by the JSP Processor. Examine the *root_dir*/logs/*server_name*/SystemOut.log file of the target application for invalid JSP directive syntax messages. Errors similar to the following in a browser indicate this kind of problem:

     ```
     Message: /filename.jsp(2,1)JSPG0076E: Missing required attribute page for jsp
                 element jsp:include
     ```

     This example indicates that line 2, column 1 of the named JavaServer Pages file is missing a mandatory attribute for the jsp:include action. Similar messages are displayed for other syntax errors.
   - Examine the target application server's `SystemErr.log` files for problems with invalid Java syntax. Errors similar to `Message: Unable to compile class for JSP` in a browser indicate this kind of problem.

     The error message output from the Javac compiler will be found in the `SystemErr.log` file. It might look like:

```
JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
int myInt=121;
        ^  1 error
```

Correct the error in the JSP file and retry the file.

- Examine the log files for the target application for problems with invalid Java syntax. Errors similar to `Message: Unable to compile class for JSP` in a browser indicate this kind of problem.

  The error message output from the Javac compiler will be found in the SystemErr.log. It might look like:

```
JSPG0091E: An error occurred at line: 2 in the file: /myJsp.jsp
JSPG0093E: Generated servlet error: c:\WASROOT\temp\ ...
test.war\_myJsp.java:16: myInt is already defined in com.ibm.ws.jsp20._myJsp
int myInt = 122; String myString = "number is 122"; static int myStaticInt=22;
int myInt=121;
        ^  1 error
```

  Correct the error in the JSP file and retry the file.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

JavaServer Pages fail to compile when using precompile

Symptom

JavaServer Pages fail to compile during deployment through the administrative console when precompile is selected.

```
SystemErr R com.ibm.websphere.management.exception.AdminException:
ADMA0021E: Error in compiling jsps - xyz.war (rc=1)
```

Problem

JavaServer Pages fail to compile during deployment through the administrative console when precompile is selected when there is a dependency on another Java archive (JAR) file that is not available on any class path.

Suggested solution

You may use wsadmin scripting to precompile JSP files during enterprise application deployment. However if you want to use the administrative console, then compile all JSP files before packaging the application.

1. Add the dependent JAR to the application server.
   a. Click **Servers > Application servers > *server1* > Java and Process Management > Process Definition > Java Virtual Machine** in the console navigation.
   b. Add fully qualified dependent JAR in class path field.
   c. Click OK.
   d. Restart application server.

JSPG0089E: Mismatch found between page directive encoding Shift_JIS and xml prolog encoding UTF-8

Symptom

The following error appears:

```
JSP Processing Error

HTTP Error Code: 500

Error Message: /test.jsp(2,1) /test.jsp(2,1) JSPG0089E: Mismatch found between
page directive encoding Shift_JIS and xml prolog encoding UTF-8
```

Problem

The pageEncoding attribute in the jsp:directive.page element is not UTF-8.

Suggested solution

JavaServer Pages must specify a prolog that matches the encoding specified in the page directive. For example,

```
<?xml version="1.0" encoding="Shift_JIS"?>
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
<jsp:directive.page language="java" contentType="text/html";
 charset=Shift_JIS" pageEncoding="Shift_JIS"/>
<jsp:text>XXXXXjsp:text>XXXXX>
</jsp:root>
```

For additional information, see section JSP.4.1, Page Character Encoding, in the JavaServer Pages specification and section 4.3.3 and appendix F.1 of the Extensible Markup Language (XML) specification

If none of these steps solves the problem, check to see if the problem is identified and documented using the links in the topic, Diagnosing and fixing problems: Resources for learning. If you do not see a problem that resembles yours, or if the information provided does not solve your problem, contact IBM support for further assistance.

For current information available from IBM Support on known problems and their resolution, see the IBM Support page. The IBM Support page contains documents that can save you time gathering information needed to resolve this problem.

Backing up and recovering servlets

Servlet source and class files, user profile data, Hypertext Transfer Protocol (HTTP) configuration, and administrative configuration should be considered for backup when using servlets. You should consider saving your HTTP configuration because changes to the HTTP configuration are often made to enable WebSphere Application Server to serve servlets and JSP requests, and to enable WebSphere Application Server security. You should consider backing up the user profile data if you use the User Profile function of WebSphere Application Server.

Procedure

- Backup servlet source and class files. Application code and configuration such as bindings, is located by default in the *profile_root*/`installedApps` directory. By saving this directory, you save your installed applications, including HTML, servlets, JavaServer Pages (JSP) files, and enterprise beans. Normally, each application is located in a separate subdirectory, so you can choose to save all applications or a subset.
  1. Save all installed applications. The commands below have been wrapped for display purposes. Enter each as a single command.

     ```
     SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
      OBJ(('/profile_root/installedApps'))
     ```

  2. Saves the sampleApp application only. The commands below have been wrapped for display purposes. Enter each as a single command.

     ```
     SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
      OBJ(('/profile_root/installedApps/cellName/sampleApp.ear'))
     ```

If you have located utility or general purpose classes in other directories, such as *profile_root*/lib/app or *profile_root*/lib/ext, be sure to include those locations in your backup plan as well.

- Save your HTTP configuration.

  Attention: The following information applies to IBM HTTP Server for iSeries (powered by Apache). If you are using Lotus Domino HTTP Server, see the Notes.net Documentation Library.

  1. Save the HTTP server instances for IBM HTTP Server for iSeries (powered by Apache). The HTTP server instances for IBM HTTP Server for iSeries (powered by Apache) are members of the QATMHINSTC file in the library QUSRSYS. An example save command for this file could be the following: SAVOBJ OBJ(QATMHINSTC) LIB(QUSRSYS) DEV(*SAVF) OBJTYPE(*FILE) SAVF(WSALIB/WSASAVF)
  2. Save the HTTP configurations for IBM HTTP Server for iSeries (powered by Apache). The HTTP configurations for IBM HTTP Server for iSeries (powered by Apache) are stored in the integrated file system in a subdirectory, chosen when the configuration was created. The recommended location is within the WebSphere instance directory. You can determine this file location by inspecting HTTP server instance member in the QATMHINSTC file in library QUSRSYS. An example save command for this file could be the following: SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file') OBJ(('*profile_root*/profile/apache/conf') ('*profile_root*/profile/htdocs')) where profile is the name of your instance. The default instance name is default.

Backing up and recovering JavaServer Pages files

JavaServer Pages source and generated servlet classes, Hypertext Transfer Protocol (HTTP) configuration, and administrative configuration should be considered for backup when using JavaServer Pages files.

Procedure

- Save installed applications. Application code and configuration such as bindings, is located by default in the *profile_root*/installedApps directory. By saving this directory, you save your installed applications, including HTML, servlets, JavaServer Pages (JSP) files, and enterprise beans. Normally, each application is located in a separate subdirectory, so you can choose to save all applications or a subset.

  1. Save all installed applications. The command below has been wrapped for display purposes. Enter the following as a single command, with a space between the end of DEV parameter and OBJ.

     ```
     SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
        OBJ(('/profile_root/installedApps'))
     ```

  2. Save the sampleApp application only. The command below has been wrapped for display purposes. Enter the following as a single command with a space between the end of DEV parameter and OBJ.

     ```
     SAV DEV('/QSYS.lib/wsalib.lib/wsasavf.file')
        OBJ(('/profile_root/installedApps/cellName/sampleApp.ear'))
     ```

- Save and restore your JSP files. When JSP files are run, a servlet class is generated, compiled, and then run. When saving and restoring your JSP files, you can elect to save only the JSP source or the generated files as well.
  - If you save and restore only the JSP source, the servlet source and class files are regenerated when they are invoked. This is a simpler, smaller save and restore operation. Note that regeneration slows the first requests, and default optimization is done on the generated Java programs.
  - If you save and restore the source and generated files, no regeneration is done. If you have optimized Java programs to levels other than the default, this optimization is preserved.

Example

WebSphere Application Server places the generated files (*.class, .java*, and optionally, *.dat*) in a temporary directory under the WebSphere Application Server instance. For example, the default instance stores the generated files in this directory:

/profile_root/temp/*node_name/application_server/enterprise_app/web_module*

In this example:

- *profile* is the name of your instance. The default instance name is default.
- *node_name* is the name of the iSeries server or partition on which your WebSphere Application Server instance is running
- *application_server* is the name of your WebSphere Application Server
- *enterprise_app* is the name of the enterprise application to which the JSP file belongs
- *web_module* is the web module that contains your JSP file.

Attention: A `.dat` file is a helper file used by the generated servlet.

Administering RRD applications

Remote request dispatcher

Remote Request Dispatcher (RRD) is a pluggable extension to the web container for application frameworks, servlets, and JavaServer Pages to include content from outside of the current Java virtual machine (JVM) for the resource, as part of the response sent to the client.

Remote request dispatcher is an extensible infrastructure for other components and stack products to add custom extensions like generators and handlers, to the RRD extension. The remote request dispatcher extension enhances the standard Java Platform, Enterprise Edition (Java EE) `javax.servlet.RequestDispatcher` implementation to be aware of locating remote resources using web services to communicate between machines within a WebSphere Application Server, Network Deployment (ND) core group. The remote request dispatcher extension reports any errors that occur on the remote server back to the originating server. It can also use SSL for secure communications and WS-Security security context propagation between servers. See the rrdSecurity.props file topic for more information.

RRD portlet support carries forward the remote request dispatcher concept to portlets and enhances the portlet container for invocation of portlets outside of the current JVM resource.

By using the RRD extension, you can share request load across multiple machines and JVMs by including remote servers within the cell. If RRD resource is memory or processor intensive, the calling resource is not affected as much as a standard RequestDispatcher running within the same JVM. RRD solves this problem by separating resources into a different JVM.

Capabilities

- Requests on remote server are treated as include requests. Filters and request listeners are started as if the dispatch type is INCLUDE.
- Serializable request attributes and query parameters are sent to remote server.
- Security context is sent to a remote server through LTPA tokens.
- Servlet parameters and OutputStream

  Request parameters are passed to remote server.
- Response headers that are set by the remotely included resource are ignored similar to includes on a local server. Internal headers such as Set-Cookie can still be set and are propagated back.
- All original request headers are passed to remote server
  - Similar to the plug-in for WebSphere Application Server.
  - Method calls return the state as if they are on local server. For example, getServer returns the local server name or isSecure returns whether the request to the ‘local' server has been secure.
- Cookies and sessions
  - Cookies are passed to the remote server as part of headers.

  - Sessions in local and remote servers use the same cookie or session ID for a given client which is similar to includes in the same server. If a session exists on a remote server, the session cookie contains the information for both the servers to maintain the affinity to the remote server.
- Exceptions
  - If there is an exception on the remote server, the server returns an RRD-specific web services fault which wraps the original exception created by the application.
  - Attempt to recreate the original exception on the local server if the exception class exists on both servers. If the original exception cannot be recreated, an RRD-specific ServletException is constructed and used instead.
  - The exception is recreated by the local server for error handling purposes.
- Dynamic cache

  When dynamic cache is enabled, caching is performed on the local and remote machine.
- Security

  You can use SSL to encrypt RRD messages between application servers. SSL is enabled by default, however, you must also pass security context needs through RRD to ensure that the security state is available in the remote machine. RRD uses WS-Security to pass this information, but this security context propagation is disabled by default. See the rrdSecurity.props file topic for additional information.

Asynchronous request dispatching settings

Asynchronous request dispatching settings

Use this page to enable the asynchronous request dispatcher (ARD), which enables servlets and JSP pages to make standard include calls concurrently on separate threads.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Web Container Settings** > **Web container** > **Asynchronous request dispatching**.

Additionally, the initiation and the insertion of the include contents can be separated so that the include has more time to execute before it needs to be written to the response. ARD requires aggregation of the include contents with the original response. The application server can aggregate the contents in memory or the client browser can aggregate the contents through AJAX. Aggregation type is configurable at the application level.

Allow Asynchronous Request Dispatching:

Enables applications installed on this server to use asynchronous request dispatching.

Asynchronous include timeout:

Specifies the default timeout in milliseconds to complete asynchronous includes.

Data type	Integer
Units	Milliseconds
Default	60000
Range	

Maximum expired requests per minute:

Specifies the maximum percentage of expired response versus total response in one minute before switching to synchronous requests.

Data type	Integer

Units	Percentage
Default	15
Range	

Maximum memory size of results store:

Specifies the maximum size of store for client side requests.

Data type	Integer
Units	Megabytes
Default	100
Range	

Asynchronous request dispatching settings

Asynchronous request dispatching settings

Use this page to enable the asynchronous request dispatcher (ARD), which enables servlets and JSP pages to make standard include calls concurrently on separate threads.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Web Container Settings** > **Web container** > **Asynchronous request dispatching**.

Additionally, the initiation and the insertion of the include contents can be separated so that the include has more time to execute before it needs to be written to the response. ARD requires aggregation of the include contents with the original response. The application server can aggregate the contents in memory or the client browser can aggregate the contents through AJAX. Aggregation type is configurable at the application level.

Allow Asynchronous Request Dispatching

Enables applications installed on this server to use asynchronous request dispatching.

Asynchronous include timeout

Specifies the default timeout in milliseconds to complete asynchronous includes.

Data type	Integer
Units	Milliseconds
Default	60000
Range	

Maximum expired requests per minute

Specifies the maximum percentage of expired response versus total response in one minute before switching to synchronous requests.

Data type	Integer
Units	Percentage
Default	15
Range	

Maximum memory size of results store

Specifies the maximum size of store for client side requests.

Data type	Integer
Units	Megabytes
Default	100
Range	

Administering RRD applications

Remote request dispatcher

Remote Request Dispatcher (RRD) is a pluggable extension to the web container for application frameworks, servlets, and JavaServer Pages to include content from outside of the current Java virtual machine (JVM) for the resource, as part of the response sent to the client.

Remote request dispatcher is an extensible infrastructure for other components and stack products to add custom extensions like generators and handlers, to the RRD extension. The remote request dispatcher extension enhances the standard Java Platform, Enterprise Edition (Java EE) `javax.servlet.RequestDispatcher` implementation to be aware of locating remote resources using web services to communicate between machines within a WebSphere Application Server, Network Deployment (ND) core group. The remote request dispatcher extension reports any errors that occur on the remote server back to the originating server. It can also use SSL for secure communications and WS-Security security context propagation between servers. See the rrdSecurity.props file topic for more information.

RRD portlet support carries forward the remote request dispatcher concept to portlets and enhances the portlet container for invocation of portlets outside of the current JVM resource.

By using the RRD extension, you can share request load across multiple machines and JVMs by including remote servers within the cell. If RRD resource is memory or processor intensive, the calling resource is not affected as much as a standard RequestDispatcher running within the same JVM. RRD solves this problem by separating resources into a different JVM.

Capabilities

- Requests on remote server are treated as include requests. Filters and request listeners are started as if the dispatch type is INCLUDE.
- Serializable request attributes and query parameters are sent to remote server.
- Security context is sent to a remote server through LTPA tokens.
- Servlet parameters and OutputStream

  Request parameters are passed to remote server.
- Response headers that are set by the remotely included resource are ignored similar to includes on a local server. Internal headers such as Set-Cookie can still be set and are propagated back.
- All original request headers are passed to remote server
  - Similar to the plug-in for WebSphere Application Server.
  - Method calls return the state as if they are on local server. For example, getServer returns the local server name or isSecure returns whether the request to the ‘local' server has been secure.
- Cookies and sessions
  - Cookies are passed to the remote server as part of headers.
  - Sessions in local and remote servers use the same cookie or session ID for a given client which is similar to includes in the same server. If a session exists on a remote server, the session cookie contains the information for both the servers to maintain the affinity to the remote server.
- Exceptions

- If there is an exception on the remote server, the server returns an RRD-specific web services fault which wraps the original exception created by the application.
- Attempt to recreate the original exception on the local server if the exception class exists on both servers. If the original exception cannot be recreated, an RRD-specific ServletException is constructed and used instead.
- The exception is recreated by the local server for error handling purposes.

- Dynamic cache

  When dynamic cache is enabled, caching is performed on the local and remote machine.
- Security

  You can use SSL to encrypt RRD messages between application servers. SSL is enabled by default, however, you must also pass security context needs through RRD to ensure that the security state is available in the remote machine. RRD uses WS-Security to pass this information, but this security context propagation is disabled by default. See the rrdSecurity.props file topic for additional information.

Configuring HTTP sessions

Configuring session management by level

When you configure session management at the web container level, all applications and the respective web modules in the web container normally inherit that configuration, setting up a basic default configuration for the applications and web modules below it. However, you can set up different configurations individually for specific applications and web modules that vary from the web container default. These different configurations override the default for these applications and web modules only.

About this task

An enterprise application can contain web modules, and so can an OSGi application. At the web container level and at the application level, the process for configuring HTTP sessions is the same whether the web module is part of an enterprise application or an OSGi application. For enterprise applications only, you can also configure HTTP sessions at the web module level.

Note: When you overwrite the default session management settings at the application level, all the web modules below the application inherit the new setting unless they too are set to overwrite these settings.

Note: Session management configuration is a post-deployment configuration and is tied to existing targets. If you change the target mapping after you configure session management, you must return to the session management configuration page in the administrative console or use wsadmin scripting and apply the changes. Apply the changes to module targets if session management is configured for a web module. Apply the changes to all targets if session management is configured for an application level.

Procedure

1. Open the administrative console.
2. Change the configuration for the web container, enterprise application, or web module level. The following substeps explain how to access the configuration panels for each of these levels.
   - Complete the following substeps for the web container level:
     a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
     b. Under **Container Settings**, expand **Web Container Settings** and click **Web container**.
     c. Under **Additional Properties**, click **Session management**.
   - Complete the following substeps for the application level:

     If the application is an enterprise application:
     a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.

b. Under **Web Module Properties**, click **Session management**.

If the application is an OSGi application:

a. Click **Applications** > **Application Types** > **Business-level applications** > ***application_name*** > ***eba_asset_name***.

b. Under **Additional Properties**, click **Session management**.

- Complete the following substeps for the web module level, if the application is an enterprise application:
  a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
  b. Under **Modules**, click **Manage Modules** > ***module_name***.
  c. Under Additional Properties, click **Session Management**.

3. Manage sessions by configuring session tracking, session timeouts and serializing access.
4. If you are working at the web module or application level and want these settings to override the inherited Session Management settings, under **General Properties** select **Override session management**.
5. Click **Apply** and **Save**.

Session management settings

Use this page to manage HTTP session support. This support includes specifying a session tracking mechanism, setting maximum in-memory session count, controlling overflow, and configuring session timeout.

To view this administrative console page at the web container level, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Session management**.

Important:
You can override the session management settings at the application level.

Session tracking mechanism:

Table 239. HTTP session tracking mechanism. This table describes mechanisms for HTTP session management.

Mechanism	Function	Default
Enable SSL ID tracking	Specifies that session tracking uses Secure Sockets Layer (SSL) information as a session ID. Enabling SSL tracking takes precedence over cookie-based session tracking and URL rewriting. There are two parameters available if you enable SSL ID tracking: SSLV3Timeout and Secure Authentication Service (SAS). SSLV3Timeout specifies the time interval after which SSL sessions are renegotiated. This parameter is a high setting and modification does not provide any significant impact on performance. The SAS parameter establishes an SSL connection only if it goes out of the Java Virtual Machine (JVM) to another JVM. If all the beans are co-located within the same JVM, the SSL used by SAS does not hinder performance. These parameters are set by editing the `sas.server.properties` and `sas.client.props` files, located in the *product_installation_root*`/properties` directory, where *product_installation_root* is the directory where WebSphere Application Server is installed. **Important:** SAS is supported only between Version 6.0.x and previous version servers that have been federated in a Version 6.1 cell.	false (cleared)
Enable cookies	Specifies that session tracking uses cookies to carry session IDs. If cookies are enabled, session tracking recognizes session IDs that arrive as cookies and tries to use cookies for sending session IDs. If cookies are not enabled, session tracking uses Uniform Resource Identifier (URL) rewriting instead of cookies (if URL rewriting is enabled). Enabling cookies takes precedence over URL rewriting. Click **Enable cookies** to change these settings. Application level session management settings override the server level session management settings. Because session management is defined at the application level, enabling cookies for the administration console is handled in the `deployment.xml` file.	true (selected)

Table 239. HTTP session tracking mechanism (continued). This table describes mechanisms for HTTP session management.

Mechanism	Function	Default
Enable URL rewriting	Specifies that the session management facility uses rewritten URLs to carry the session IDs. If URL rewriting is enabled, the session management facility recognizes session IDs that arrive in the URL if the encodeURL method is called in the servlet.	false (cleared)
Enable protocol switch rewriting	This option is only available when **Enable URL rewriting** is selected. This option specifies that the session ID is added to a URL when the URL requires a switch from HTTP to HTTPS or from HTTPS to HTTP. If rewriting is enabled, the session ID is required to go between HTTP and HTTPS.	false (cleared)

Maximum in-memory session count:

Specifies the maximum number of sessions to maintain in memory for each web module.

The meaning differs depending on whether you are using in-memory or distributed sessions. For in-memory sessions, this value specifies the number of sessions in the base session table for a web module. Use the **Allow overflow** property to specify whether to limit sessions to this number for the entire session management facility or to allow additional sessions to be stored in secondary tables. For distributed sessions, this value specifies the size of the memory cache for sessions of each web module. When the session cache has reached its maximum size and a new session is requested, the session management facility removes the least recently used session from the cache to make room for the new one.

Note: Do not set this value to a number less than the maximum thread pool size for your server.

Allow overflow:

Specifies that the number of sessions in memory can exceed the value specified by the Max in-memory session count property. This option is valid only in non-distributed sessions mode.

Session timeout:

Specifies how long a session can go unused before it is no longer valid. Specify either `Set timeout` or `No timeout`. Specify the value in minutes greater than or equal to two.

The value specified in a web module deployment descriptor file takes precedence over the administrative console settings. However, the value of this setting is used as a default when the session timeout is not specified in a web module deployment descriptor. To preserve performance, the invalidation timer is not accurate to the second. When the write frequency is time-based, ensure that this value is least twice as large as the write interval.

Security integration:

Specifies that when security integration is enabled, the session management facility associates the identity of users with their HTTP sessions. Session security (security integration) is enabled by default.

Serialize session access:

Specifies that concurrent session access in a given server is not permitted.

Table 240. Serialize session access. This table describes mechanisms for serialize session access.

Mechanism	Function
Maximum wait time	Specifies the maximum amount of time a servlet request waits on an HTTP session before starting. This parameter is optional and expressed in seconds. The default is five seconds. Under normal conditions, a servlet request waits for access to an HTTP session and is notified by the request that currently owns the given HTTP session when the request finishes.

Table 240. Serialize session access (continued). This table describes mechanisms for serialize session access.

Mechanism	Function
Allow access on timeout	Specifies whether the servlet starts normally or stops processing from a timeout. The servlet starts normally when this box is checked. If this box is not checked, the servlet stops processing and error logs are generated.

Session recovery support

For session recovery support, WebSphere Application Server provides distributed session support in the form of database sessions. You can use session recovery support when the user's session data must be maintained across a server restart or when the user's session data is too valuable to lose through an unexpected server failure.

All the attributes set in a session must implement java.io.Serializable if the session requires external storage. In general, consider making all objects held by a session serialized, even if immediate plans do not call for session recovery support. If the website grows, and session recovery support becomes necessary, the transition occurs transparently to the application if the sessions only hold serialized objects. If not, a switch to session recovery support requires coding changes to make the session contents serialized.

Configuring session tracking

About this task

Review the Session tracking options to plan your approach to session management. To configure session tracking, complete the following:

Procedure

1. Go to the appropriate level of Session Management.
2. Specify the session tracking mechanism that you want to pass the session ID between the browser and the servlet:
   - To track sessions with cookies, click **Enable Cookies**.

     To change the cookie settings, click **Modify**.
   - To track sessions with URL rewriting, click **Enable URL Rewriting**.

     If you want to enable protocol switch rewriting, click **Enable protocol switch rewriting.**
   - To track sessions with SSL information, click **Enable SSL ID tracking**.

     Note: Session tracking using the SSL ID is deprecated in WebSphere Application Server version 7.0. You can reconfigure session tracking to use cookies or modify the application to use URL rewriting.
3. Click **Apply**.
4. Click **Save**.

Session tracking options

HTTP session support also involves session tracking. You can use cookies, URL rewriting, or Secure Sockets Layer (SSL) information for session tracking.

the following tracking methods are available:
- Session tracking with cookies
- Session tracking with URL rewriting
- Session tracking with Secure Sockets Layer (SSL) information

Session tracking with cookies: Tracking sessions with cookies is the default. No special programming is required to track sessions with cookies.

Session tracking with URL rewriting: An application that uses URL rewriting to track sessions must adhere to certain programming guidelines. The application developer needs to do the following:
- Program servlets to encode URLs
- Supply a servlet or JavaServer Pages (JSP) file as an entry point to the application

Using URL rewriting also requires that you enable URL rewriting in the session management facility.

Note: In certain cases, clients cannot accept cookies. Therefore, you cannot use cookies as a session tracking mechanism. Applications can use URL rewriting as a substitute.

Program session servlets to encode URLs

Depending on whether the servlet is returning URLs to the browser or redirecting them, include either the encodeURL method or the encodeRedirectURL method in the servlet code. Examples demonstrating what to replace in your current servlet code follow.

Rewrite URLs to return to the browser

Suppose you currently have this statement:

```
out.println("<a href=\"/store/catalog\">catalog<a>");
```

Change the servlet to call the encodeURL method before sending the URL to the output stream:

```
out.println("<a href=\"");
out.println(response.encodeURL ("/store/catalog"));
out.println("\">catalog</a>");
```

Rewrite URLs to redirect

Suppose you currently have the following statement:

```
response.sendRedirect ("http://myhost/store/catalog");
```

Change the servlet to call the encodeRedirectURL method before sending the URL to the output stream:

```
response.sendRedirect (response.encodeRedirectURL ("http://myhost/store/catalog"));
```

The encodeURL method and encodeRedirectURL method are part of the HttpServletResponse object. These calls check to see if URL rewriting is configured before encoding the URL. If it is not configured, the calls return the original URL.

You can also configure session support to enable protocol switch rewriting. When this option is enabled, the product encodes the URL with the session ID for switching between HTTP and HTTPS protocols.

gotcha: If you want to encode a URL, you must enable the AlwaysEncodeURL custom property and set the value to true.

Session management properties, like the session management configuration, can be configured at the server, application, or web module level. The following steps are for setting the custom properties for session management at the server level.

1. In the administrative console click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Session management**.
2. Under **Additional Properties** select **Custom Properties**.
3. On the Custom Properties page, click **New**.
4. On the settings page, enter the property that you want to configure in the **Name** field and the value that you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.

6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

Supply a servlet or JSP file as an entry point

The entry point to an application, such as the initial screen presented, may not require the use of sessions. However, if the application in general requires session support (meaning some part of it, such as a servlet, requires session support), then after a session is created, all URLs are encoded to perpetuate the session ID for the servlet (or other application component) requiring the session support.

The following example shows how you can embed Java code within a JSP file:

```
<%
response.encodeURL ("/store/catalog");
%>
```

Session tracking with SSL information (Deprecated):

Note: Session tracking using the SSL ID is deprecated in WebSphere Application Server version 7.0. You can configure session tracking to use cookies or URL rewriting.

No special programming is required to track sessions with Secure Sockets Layer (SSL) information.

To use SSL information, turn on **Enable SSL ID tracking** in the session management property page. Because the SSL session ID is negotiated between the web browser and HTTP server, this ID cannot survive an HTTP server failure. However, the failure of an application server does not affect the SSL session ID if an external HTTP server is present between WebSphere Application Server and the browser.

SSL tracking is supported for the IBM HTTP Server and iPlanet Web Servers only. You can control the lifetime of an SSL session ID by configuring options in the web server. For example, in the IBM HTTP Server, set the configuration variable SSLV3TIMEOUT to provide an adequate lifetime for the SSL session ID. An interval that is too short can cause a premature termination of a session. Also, some web browsers might have their own timers that affect the lifetime of the SSL session ID. These web browsers may not leave the SSL session ID active long enough to serve as a useful mechanism for session tracking. The internal HTTP Server of WebSphere Application Server also supports SSL tracking.

When using the SSL session ID as the session tracking mechanism in a cloned environment, use either cookies or URL rewriting to maintain session affinity. The cookie or rewritten URL contains session affinity information that enables the web server to properly route a session back to the same server for each request.

Serializing access to session data

The Servlet API supports concurrent access to a session in a given server instance. WebSphere Application Server provides an option to prevent the concurrent access to a session in a given server instance so that concurrent modification of a session does not occur in a given server instance.

About this task

Preventing concurrent access to a session is achieved by synchronizing the requests based on session. When this feature is turned on, a session is obtained for the request before invoking the servlet and requests are synchronized by locking the session for the servlet initiation time. Note that synchronization is based on the memory copy of session. So this feature cannot serialize requests across servers based on session when session affinity fails.

You can also use the serializing access to session data feature to synchronize session objects inside of servlets or JavaServer pages. Applications cannot synchronize session objects inside of their servlets or JavaServer Pages because a deadlock with the session manager may occur. The deadlock occurs

because the session manager does not expect the use of more than one locking mechanism. You can ensure that only one request can access the session at a time through the use of the configuration option, Allow serial access.

Use this feature only when concurrent modification of the same session data is possible and is not desirable by the application. This feature has overhead of serializing the requests based on a session.

Do the following to synchronize session access:

Procedure

1. Select the level of Session Management on which you want to serialize session access.
2. Under Serialize Session access, click **Allow serial access**.
3. In the Maximum wait time box, type the amount of time, in milliseconds, a servlet waits on a session before continuing execution. The default is 120000 milliseconds or two minutes.
4. Select **Allow access on timeout** if you want the servlet to gain access to the session and continue normal execution even if the session is still locked by another servlet. If you do not select this box, the servlet execution will abort when the session request times out.
5. Click **Apply**.
6. Click **Save**.

Cookie settings

Use this page to configure cookie settings for session management.

To view this administrative console page, click **Servers** > **Server types** > **WebSphere application servers** > ***server_name*** > **Session management** > **Enable cookies**.

Cookie name:

Specifies a unique name for the session management cookie. The servlet specification requires the name JSESSIONID. However, for flexibility, you can configure this value.

Restrict cookies to HTTPS sessions:

Specifies that the session cookies include the secure field. Enabling this feature restricts the exchange of cookies to HTTPS sessions only.

Set cookies as HTTP only to help prevent cross-site scripting attacks:

Specifies that session cookies include the HTTP only field. When checked, browsers that support the HTTP only attribute do not enable cookies to be accessed by client-side scripts. For security cookies, see the global security settings for web single sign-on (SSO).

Cookie domain:

Specifies the domain field of a session tracking cookie. This value controls whether a browser sends a cookie to particular servers. For example, if you specify a particular domain, session cookies are sent to hosts in that domain. The default domain is the server.

Cookie maximum age:

Specifies the amount of time that the cookie lives on the client browser. Specify that the cookie lives only as long as the current browser session, or to a maximum age. If you choose the maximum age option, specify the age in seconds. This value corresponds to the Time to Live (TTL) value described in the Cookie specification.

Default is the current browser session which is equivalent to setting the value to -1.

Cookie path:

Specifies that a cookie is sent to the URL designated in the path. Specify any string representing a path on the server. A slash (/) indicates root directory. Specify a value to restrict the paths to which the cookie is sent. By restricting paths, you prevent the cookie from going to certain URLs on the server. If you specify the root directory, the cookie is sent no matter which path on the given server is accessed.

Set the cookie path to match the context root for each application. This setting restricts the cookie from being sent to other applications and results in having different cookies created when accessing multiple applications.

Session management custom properties

You can specify additional settings for session management through setting custom properties.

Session management properties, like the session management configuration, can be configured at the server, application, or web module level. The following steps are for setting the custom properties for session management at the server level.

1. In the administrative console click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Session management**.
2. Under **Additional Properties** select **Custom Properties**.
3. On the Custom Properties page, click **New**.
4. On the settings page, enter the property that you want to configure in the **Name** field and the value that you want to set it to in the **Value** field.
5. Click **Apply** or **OK**.
6. Click **Save** on the console task bar to save your configuration changes.
7. Restart the server.

You can use the custom properties page to define the following session management properties:

- “AlwaysEncodeURL” on page 2584
- “CloneSeparator” on page 2584
- “CloneSeparatorChange” on page 2584
- “DebugSessionCrossover” on page 2584
- “ForceSessionInvalidationMultiple” on page 2584
- “HideSessionValues” on page 2585
- “HttpSessionCloneId” on page 2585
- “HttpSessionIdLength” on page 2585
- “HttpSessionIdReuse” on page 2585
- “HttpSessionReaperPollInterval” on page 2585
- “NoAdditionalSessionInfo” on page 2586
- “OptimizeCacheIdIncrements” on page 2586
- “SecurityUserIgnoreCase” on page 2586
- “Servlet21SessionCompatibility” on page 2586
- “SessionIdentifierMaxLength” on page 2586
- “SessionRewriteIdentifier” on page 2587
- “SessionTableName” on page 2587
- “UseInvalidatedId” on page 2587
- “UseOracleBLOB” on page 2587

- “UsingApplicationSessionsAndInvalidateAll” on page 2587
- “UsingCustomSchemaName” on page 2587

AlwaysEncodeURL:

The Servlet 2.5 specification specifies to not encode the URL on a response.encodeURL call if it is not necessary. To support backward compatibility when URL encoding is enabled, set the *AlwaysEncodeURL* custom property to `true` to call the encodeURL method. The URL is always encoded, even if the browser supports cookies.

CloneSeparator:

Use this property to specify a different character as the clone separator in session cookies. The value specified for this custom property must be a single character.

This property was set as a web container custom property in version 6.1 but must now be set as a session management custom property.

bprac: This property should only be used as a means to provide more flexibility if you have a situation where you cannot use either a colon (:), or a plus sign (+) as the clone separator in session cookies. You should understand the clone character requirements of other products running on your system before using this property to change the clone separator character.

The fact that any character can be specified as the value for this custom property does not imply that the character you specify will function correctly. This fact also does not imply that IBM is responsible for fixing any problem that might arise from using an alternative character.

CloneSeparatorChange:

Use this property to maintain session affinity. The clone ID of the server is appended to session identifier separated by colon. On some Wireless Application Protocol (WAP) devices, a colon is not allowed. Set this property to `true` to change clone separator to a plus sign (+).

DebugSessionCrossover:

The *DebugSessionCrossover* custom property enables code to perform additional checks to verify that only the session associated with the request is accessed or referenced. Messages are logged if any discrepancies are detected.

Set this property to `true` to enable session data crossover detection.

See article, HTTP session problems, for additional information.

ForceSessionInvalidationMultiple:

The *ForceSessionInvalidationMultiple* custom property indicates whether the session manager should wait indefinitely for a request to complete before attempting to invalidate the session, or should attempt to invalidate a session after the specified time limit has elapsed. The default value for this property is 1.

- If you specify 0 (zero) for this custom property, the session manager waits indefinitely until a request is complete before attempting to invalidate the session.

  If your requests normally are not bound by a response time limit, specify 0 for this property.
- If you specify a positive integer, such as 1, 2, or 3, for this custom property, even if a session is not known to have completed, the session manager attempts to invalidate the session, if the indicated time period since the last access occurred has elapsed. This time period is the result of multiplying the value specified for this property and the value specified for the Session Timeout property. For example, if you

specify 2 minutes for the Session Timeout property and 2 for the ForceSessionInvalidationMultiple property, the session manager attempts to invalidate the session after 4 minutes.

If you want to invalidate your sessions after a certain amount of time has elapsed, specify the appropriate positive integer for this property.

HideSessionValues:

The *HideSessionValues* custom property prevents the logging of session attribute values in session manager traces.

Applications store these session attribute values. However, you might not want to see these values in application server traces. Set this property to `true` if you do not want to see these values in application server traces.

HttpSessionCloneId:

Use this property to change the clone ID of the cluster member. Within a cluster, this ID must be unique to maintain session affinity. When set, this name overwrites the default name generated by WebSphere Application Server.

Default clone ID length: 40

HttpSessionIdLength:

Use this property to configure the session identifier length. Do not use an extremely low value; using a low value results in reduced number of combinations possible, thereby increasing risk of guessing the session identifier. In a cluster, all cluster members should be configured with same ID length. Allowed range: 8 to 128. Default length: 23.

HttpSessionIdReuse:

The custom property *HttpSessionIdReuse* determines whether the session manager can use the session ID sent from a browser to preserve session data across web applications that are running in an environment that is not configured for session persistence.

In a multi-JVM environment that is not configured for session persistence setting this property to `true` enables the session manager to use the same session information for all of a user's requests even if the web applications that are handling these requests are governed by different JVMs. The default value for this property is `false`. Set this property to `true` if you want to enable the session manager to use the session ID sent from a browser to preserve session data across web applications that are running in an environment that is not configured for session persistence.

HttpSessionReaperPollInterval:

Use this property to specify, in seconds, a wake-up interval for the process that removes invalid sessions. The value specified for this property overrides the default installation value, which is between 30 and 360 seconds, and ensures that the reaper process runs at a specific interval.

If the maximum inactive interval is less than 2 minutes, the reaper poll interval may be as short as 30 seconds.

If the maximum inactive interval is more than 15 minutes, the reaper poll interval can be as long as 6 minutes.

Because the default timeout and maximum inactive interval is 30 minutes, the reaper interval is usually between 5 and 6 minutes.

For example, you might want to use this property if you want the installation timed out sessions invalidated more frequently than 5 to 6 minutes. Specifying HttpSessionReaperPollInterval=120 ensures that sessions are invalidated within 2 minutes of timing out.

The minimum value for this property is 30 seconds. If a value less than the minimum is entered, the specified property is ignored and an appropriate value is automatically determined and used. The maximum inactive interval is the session timeout. The default is based on maximum inactive interval set in session management.

NoAdditionalSessionInfo:

Set this value to "true" to force removal of information that is not needed in session identifiers.

OptimizeCacheIdIncrements:

Set the *OptimizeCacheIdIncrements* custom property to true to make the session manager assess whether the in-memory session for a web module is older than the copy in persistent store. Setting this property resolves the continually increasing cache ID.

If HTTP session management is configured to use session persistence and the user's browser session is moving back and forth across multiple web applications you might see extra persistent store activity as the in-memory sessions for a web module are refreshed from the persistent store. As a result, the cache IDs are continually increasing and the in-memory session attributes are overwritten by those of the persistent copy. Set this property to `true` if you want to prevent the cache IDs from continually increasing.

If the configuration is a cluster, ensure that the system times of each cluster member is identical as possible.

SecurityUserIgnoreCase:

Set this property to `true` if you want the session security identity and the client security identity to be considered a match even if their cases are different.

When a user configures session security integration, the session manager compares the security identity of th session owner with the security identity of the client request. Because the matching criteria is case sensitive, if these two identities do not exactly match, an UnauthorizedSessionRequestException is sent back to the client.

If you have situations where you want the session security identity and the client security identity to be considered a match even if their cases are different, add the SecurityUserIgnoreCase custom property to your Web container configuration settings, and set the property to `true`. When this property is set to true, an UnauthorizedSessionRequestException does not occur if the session security identity and the client security identity are identical except for their cases. For example, when this property is set to true, the session security identity USER1 matches the client security identities User1 and user1.

Servlet21SessionCompatibility:

Set this custom property to true to enable global session behavior. In Servlet 2.2 and later, sessions are scoped at the Web module level. The default is **false**.

Note: This property is deprecated. The IBMApplicationSession method replaces the function of the Servlet21SessionCompatibility custom property.

SessionIdentifierMaxLength:

Use this value to set maximum length that a session identifier can grow.

This property helps to find out the condition and take appropriate action to address servers fail-over. When this is specified, message is logged when specified maximum length is reached. Allowed value: integer.

SessionRewriteIdentifier:

Use this property to change the key used with URL rewriting. Default key: jsessionid.

SessionTableName:

Use this custom property to set the database table name. Allowed value: String. The default value is SESSIONS.

Some applications may rely on method ejbCreate(...) to have created the entity bean in the database. For such a requirement, setting the JVM property *com.ibm.websphere.ejbcontainer.allowEarlyInsert* to **true** overrides the default behavior.

UseInvalidatedId:

Set this custom property to true to reuse the incoming ID if the session with that ID was recently invalidated. This is a performance optimization because it prevents checking the persistent store. The default value is **true**.

UseOracleBLOB:

The *UseOracleBLOB* custom property creates the HTTP session database table using the Binary Large Object (BLOB) data type for the medium column. This property increases performance of persistent sessions when Oracle databases are used. Due to an Oracle restriction, BLOB support requires use of the Oracle Call Interface (OCI) database driver for more than 4000 bytes of data. You must also ensure that a new sessions table is created before the server is restarted by dropping your old sessions table or by changing the datasource definition to reference a database that does not contain a sessions table.

To create a sessions table using the BLOB data type, use the following name-value pair:

Name	Value
UseOracleBLOB	true

UsingApplicationSessionsAndInvalidateAll:

When the invalidateAllSet method is called, not all IBMApplicationSessions objects are checked. If you are using both the IBMApplicationSessions object and the invalidateAll call, use the following name-value pair:

Name	Value
UsingApplicationSessionsAndInvalidateAll	true

UsingCustomSchemaName:

Use this property to ensure that the session manager successfully detects the sessions table on subsequent server startups.

Set this custom property to `true` if you are using DB2 for sessions persistence and the customSchema property is not set to the default value in the DB2 JDBC driver.

The default value is false.

Configuring session tracking for Wireless Application Protocol (WAP) devices

Applications that run in a web container use sessions to keep track of individual users. Because most Wireless Application Protocol (WAP) devices do not support cookies, you can configure WAP devices to use URL rewriting to track sessions.

About this task

On most WAP devices, the maximum URL length is 128 characters. With URL rewriting, a session identifier is added to the URL itself, effectively decreasing the space available for the actual URL and the number of parameters that can be sent on a request.

To reduce the length of session identifier, you can configure key (jsessionid), session ID length and clone ID. To make these configuration changes, complete the following steps.

Procedure

1. Open the administrative console.
2. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Web Container Settings** > **Web container**.
3. Under Additional Properties, click **Custom properties**.
4. Add the appropriate properties from the following list:
   - HttpSessionIdLength
   - SessionRewriteIdentifier
   - HttpSessionCloneId
   - CloneSeparatorChange
   - NoAdditionalSessionInfo
   - SessionIdentifierMaxLength
5. Click **Apply** and **Save**.

Configuring for database session persistence

You can configure a database to collect session data for database session persistence.

About this task

To configure the session management facility for database session persistence, complete the following steps.

Procedure

1. Create and configure a JDBC provider.
2. Create a data source pointing to a database.

   Use the JDBC provider that you defined: **Resources > JDBC > JDBC Providers >** *JDBC_provider* **> Data Sources > New**. The data source should be non-JTA, for example, non-XA enabled. Note the JNDI name of the data source.

   Point to an existing database.
3. Verify that the correct database is listed under **Resources > JDBC Providers >** *JDBC_provider* **> Data Sources >** *datasource_name*. If necessary, contact your database administrator to verify the correct database name.
4. Go to the appropriate level of Session Management.
5. Under Additional Properties, click **Distributed Environment Settings**
6. Select and click **Database**.

7. Specify the Data Source JNDI name from a previous step. The database user ID and password are case-sensitive.
8. Specify the database user ID and password that is used to access the database and for table creation. When you created your data source, you might have specified a Container Managed Authentication Alias or a Component Managed Authentication Alias; however, these two settings are not used by the session manager for session persistence. The session manager uses the userId and password specified in this step for session persistence.
9. Optional: Append the schema name in the session User ID field if you want to have more than one instance of the session table.

   The session manager uses the schema name to qualify the session table name for all database operations. If only the userid is specified without the schema name, the schema name defaults to NULL and therefore a table name with NULL as the schema name, for example, NULL.SESSIONS, is created. You can create multiple session tables with different schema names, other than NULL, and access them separately by modifying the user name to contain the appropriate schema name.
10. Retype the password for confirmation.
11. Configure a table space and page sizes for DB2 session databases.
12. Switch to a multirow schema.
13. Click **OK**.
14. If you want to change the tuning parameters, click **Custom Tuning Parameters** under Additional properties.
15. Click **Apply**.
16. Click **Save**.

Switching to a multi-row schema

The multi-row schema configuration supports storing an unlimited amount of data that is only bounded by the database capacities in an application. The application can read individual fields instead of the whole record, which can help to improve performance by avoiding unnecessary Java object serialization. Configure the session management facility to store each attribute in a session object in its own row in the database by using the multi-row schema configuration.

About this task

The only practical limit that remains is the size of the session attribute object. The multi-row schema potentially has performance benefits in certain usage scenarios, such as when larger amounts of data are stored in the session but only small amounts are specifically accessed during a given servlet processing of an HTTP request. In such a scenario, avoiding unneeded Java object serialization is beneficial to performance.

In addition to allowing larger session records, using multi-row schema can yield performance benefits. However, it requires a little work to switch from single-row to multi-row schema, as shown in the following table.

By default, a single session maps to a single row in the database table used to hold sessions. With this setup, there are hard limits to the amount of user-defined, application-specific data that WebSphere Application Server can access.

Consider configuring direct single-row usage to one database and multi-row usage to another database while you verify which option suits your application needs. Do this in code by switching the data source used; then monitor performance.

Table 241. Choosing a single-row or multi-row schema configuration. Single-row or multi-row schema configuration

Programming issue	Application scenario
Reasons to use single-row	• You can read or write all values with just one record read and write. • This takes up less space in a database because you are guaranteed that each session is only one record long.
Reasons **not** to use single-row	2-megabyte limit of stored data per session.
Reasons to use multi-row	• The application can store an unlimited amount of data; that is, you are limited only by the size of the database and a 2-megabyte-per-record limit. • The application can read individual fields instead of the whole record. When large amounts of data are stored in the session but only small amounts are specifically accessed during servlet processing of an HTTP request, multi-row sessions can improve performance by avoiding unneeded Java object serialization.
Reasons **not** to use multi-row	If data is small in size, you probably do not want the extra overhead of multiple row reads when you can store everything in one row.

In the case of multi-row usage, design your application data objects not to have references to each other, to prevent circular references. For example, suppose you are storing two objects A and B in the session using HttpSession.put(..) method, and A contains a reference to B. In the multi-row case, because objects are stored in different rows of the database, when objects A and B are retrieved later, the object graph between A and B is different than stored. A and B behave as independent objects.

Procedure

1. Modify the Session Management facility properties to switch from single to multi-row schema.
2. Manually drop the table.

   To drop the table:
   a. Determine which data source configuration Session Management is using.
   b. In the data source configuration, look up the database name.
   c. Use the database facilities to connect to the database.
   d. Drop the SESSIONS table.

Configuring tablespace and page sizes for DB2 session databases

If you are using DB2 for session persistence, you can increase the page size to optimize performance for writing large amounts of data to the database. Page sizes of 8K, 16K, and 32K are supported.

About this task

To use a page size other than the default (4K), complete the following steps.

Procedure

1. If the SESSIONS table already exists, drop it from the DB2 database.
2. Create a new DB2 buffer pool and table space, specifying the same page size (8K, 16K or 32K) for both, and assign the new buffer pool to this table space.

```
DB2 Connect to session
DB2 CREATE BUFFERPOOL sessionBP SIZE 1000 PAGESIZE 8K
DB2 Connect reset
```

```
DB2 Connect to session
DB2 CREATE TABLESPACE sessionTS PAGESIZE 8K MANAGED BY SYSTEM
    USING ('D:\DB2\NODE0000\SQL00005\sessionTS.0') BUFFERPOOL sessionBP
DB2 Connect reset
```

Refer to DB2 product documentation for details.

3. Configure the correct table space name and page size in the Session Management facility. Page size is referred to as *row size* on the Session Management page.)

Results

When the product is restarted, the Session Management facility creates the new SESSIONS table in the specified tablespace based on the indicated page size.

Creating a table for session persistence

You can use a database table to collect and store session data. If you are using a database table for session persistence, you must create and define a database table that is associated with the application server.

About this task

Whenever the session manager is set for database persistence, the session manager creates a table for its use. If you want to expand the column size limits to make it more appropriate for your website, you can create the table externally. If the external table is specified as the target table in the session manager database persistence configuration, then during the session manager start up, the external table is used. In most cases it is better to let the session manager create the table during startup.

To create a table for collecting session data, do the following:

Procedure

Have your administrator create a database table for storing your session data using one of the following data definition language (DDL): For DB2:

```
CREATE TABLE <SchemaName>.sessions  (
  ID               VARCHAR(128) NOT NULL ,
  PROPID           VARCHAR(128) NOT NULL ,
  APPNAME          VARCHAR(128) NOT NULL,
  LISTENERCNT      SMALLINT ,
  LASTACCESS       BIGINT,
  CREATIONTIME     BIGINT,
  MAXINACTIVETIME  INTEGER ,
  USERNAME         VARCHAR(256) ,
  SMALL            VARCHAR(3122)  FOR BIT DATA ,
  MEDIUM           LONG VARCHAR FOR BIT DATA ,
  LARGE            BLOB(2M)
    )
```

For Oracle:

```
CREATE TABLE SESSIONS  (
  ID               VARCHAR(128) NOT NULL ,
  PROPID           VARCHAR(128) NOT NULL ,
  APPNAME          VARCHAR(128) NOT NULL,
  LISTENERCNT      SMALLINT ,
  LASTACCESS       INTEGER,
  CREATIONTIME     INTEGER,
  MAXINACTIVETIME  INTEGER ,
  USERNAME         VARCHAR(256) ,
  SMALL            RAW(2000),
  MEDIUM           LONG RAW ,
  LARGE            RAW(1)
  )
```

If the web container custom property UseOracleBLOB is set to true then:

```
CREATE TABLE SESSIONS  (
  ID              VARCHAR(128) NOT NULL ,
  PROPID          VARCHAR(128) NOT NULL ,
  APPNAME         VARCHAR(128) NOT NULL,
  LISTENERCNT     SMALLINT ,
  LASTACCESS      INTEGER,
  CREATIONTIME    INTEGER,
  MAXINACTIVETIME INTEGER ,
  USERNAME        VARCHAR(256) ,
  SMALL           RAW(2000),
  MEDIUM          BLOB,
  LARGE           RAW(1)
  )
```

Attention:

1. At run time, the session manager accesses the target table using the identity of the Java Platform, Enterprise Edition (Java EE) server in which the owning web application is deployed. Any web container that is configured to use persistent sessions must have both read and update access to the subject database table.
2. HTTP session processing uses the index defined using the `CREATE INDEX` statement to avoid database deadlocks. In some situations, such as when a relatively small table size is defined for the database, DB2 may decide not to use this index. When the index is not used, database deadlocks can occur. If database deadlocks occur, see the DB2 Administration Guide for the version of DB2 you are using for recommendations on how to calculate the space required for an index and adjust the size of the tables that you are using accordingly.
3. It might be necessary to tune DB2 to make efficient use of the sessions database table and to avoid deadlocks when accessing it. Your DB2 administrator should refer to the DB2 Administration Guide for specific information about tuning the version of DB2 that you are using.
4. During run time, the session manager may create an entry in the database for each web module of the application. This row of data is used internally for session management purposes, such as in session invalidation. Do not be concerned with this entry. It can be overlooked.

Database settings

Use this page to specify the settings for database session support.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Session management** > **Distributed environment settings** > **Database**.

Datasource JNDI name:

Specifies the datasource description.

The JNDI name of the non-XA enabled datasource from which session management obtains database connections. For example, if the JNDI name of the datasource is "jdbc/sessions", specify "jdbc/sessions." The datasource represents a pool of database connections and a configuration for that pool (such as the pool size). The datasource must already exist as a configured resource in the environment.

User ID:

Specifies the user ID for database access.

Password:

Specifies the password for database access.

DB2 row size:

Specifies the table space page size configured for the sessions table, if using a DB2 database. Possible values are 4, 8, 16, and 32 kilobytes (KB). The default row size is 4KB.

The default row size is 4KB. In DB2, it can be updated to a larger value. This can help database performance in some environments. When this value is other than 4, you must specify table space name to use this property. For 4KB pages, the table space name is optional.

Table space name:

Specifies that table space to be used for the sessions table.

This value is required when the DB2 page size is other than 4KB.

Use multi row schema:

Specifies that each session data attribute is placed in a separate row in the database, allowing larger amounts of data to be stored for each session. This action can yield better performance when session attributes are very large and few changes are required to the session attributes. If use multi row schema is not enabled, instances of application data can be placed in the same row.

Configuring write contents

In session management, you can configure which session data is written to the database or to another WebSphere instance, depending on whether you are using database persistent sessions or memory to memory replication. You can either write only session data properties that have been updated through setAttribute method and removeAttribute method calls or you can write all session data properties.

About this task

This flexibility allows for fewer code changes for the JavaServer Pages (JSP) writer when the application will be operating in a clustered environment. The following options are available in Session Management for tuning what is written back:

- Write changed (the default) - Write only session data properties that have been updated through setAttribute method and removeAttribute method calls.
- Write all - Write all session data properties.

The **Write all** setting might benefit servlet and JSP writers who change Java objects' states that reside as attributes in HttpSession and do not call HttpSession.setAttribute method.

However, the use of **Write all** could result in more data being written back than is necessary. If this situation applies to you, consider combining the use of **Write all** with **Time-based write** to boost performance overall. As always, be sure to evaluate the advantages and disadvantages for your installation.

With either Write Contents setting, when a session is first created, complete session information is written, including all of the objects bound to the session.

Table 242. Single-row or multi-row schemas. When using database session persistence, in subsequent session requests, what is written to the database depends on whether a single-row or multi-row schema has been set for the session database, as shown in the following table.

Write Contents setting	Behavior with single-row schema	Behavior with multirow schema
Write changed	If any session attribute is updated, all objects bound to the session are written.	Only the session data modified through setAttribute method or removeAttribute method calls is written.

Table 242. Single-row or multi-row schemas (continued). When using database session persistence, in subsequent session requests, what is written to the database depends on whether a single-row or multi-row schema has been set for the session database, as shown in the following table.

Write Contents setting	Behavior with single-row schema	Behavior with multirow schema
Write all	All bound session attributes are written.	All session attributes that currently reside in the cache are written. If the session has never left the cache, all session attributes are written.

Procedure

1. Go to the appropriate level of Session Management. See the *Administering applications and their environment* PDF for more information.
2. Click Distributed environment settings
3. Click Custom tuning parameters.
4. Click Custom settings.
5. Select the appropriate write contents setting.
6. Click **OK**.

Configuring write frequency

In the session management facility, you can configure the frequency for writing session data to the database or to a WebSphere instance, depending on whether you use database distributed sessions or memory-to-memory replication. You can write session data using the end of service servlet, manual update or time based update options.

About this task

This flexibility enables you to weigh session performance gains against varying degrees of failover support. The following options are available in the Session Management facility for tuning write frequency:

- **End of service servlet**- Write session data at the end of the servlet service method call.
- **Manual update**- Write session data only when the servlet calls the IBMSession.sync method.
- **Time based** (the default) - Write session data at periodic intervals, in seconds (called the *write interval*).

When a session is first created, session information is always written at the end of the service call.

If you are using the time based write option, when a server shuts down, the server typically attempts to write the latest session data, that is in the session cache, to the persistent store, as part of the shutdown process.

Using the time based write or manual update options can result in the loss of data in failover scenarios because the backup copy of the session in the persistent store, such as a database, or another JVM, might not be in sync with the session in the session cache.

Procedure

1. Go to the appropriate level of Session Management. See the *Administering applications and their environment* PDF for more information.
2. Click Distributed environment settings
3. Click Custom tuning parameters.
4. Click Custom settings.
5. Select the method of write frequency that you want to use.
6. Click **OK**.

Chapter 29. Administering web services

This page provides a starting point for finding information about web services.

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. They implement a services oriented architecture (SOA), which supports the connecting or sharing of resources and data in a very flexible and standardized manner. Services are described and organized to support their dynamic, automated discovery and reuse.

Planning to use web services

You can plan to develop and implement web services based on a variety of Java programming models.

Before you begin

The JAX-WS Web services samples demonstrate the simple message exchange patterns using both synchronous and asynchronous invocation of web services in SOAP 1.1 and SOAP 1.2 environments. The samples are composed with web service standards such as WS-Addressing (WS-A) , WS-Reliable Messaging (WS-RM), and WS-Secure Conversation (WS-SC), which you can use to complete a broad range of interoperability tests. The samples demonstrate the use of JavaBeans artifacts and static service endpoints and proxy-based clients. Additionally, a sample is provided that demonstrates Message Transmission Optimization Mechanism (MTOM).

About this task

best-practices: IBM WebSphere Application Server supports the Java API for XML-Based Web Services (JAX-WS) programming model and the Java API for XML-based RPC (JAX-RPC) programming model. JAX-WS is the next generation web services programming model extending the foundation provided by the JAX-RPC programming model. Using the strategic JAX-WS programming model, development of web services and clients is simplified through support of a standards-based annotations model. Although the JAX-RPC programming model and applications are still supported, take advantage of the easy-to-implement JAX-WS programming model to develop new web services applications and clients.

You must re-write existing JAX-RPC applications if you want to take advantage of the features of the JAX-WS programming model.

Web services reflect the service-oriented architecture approach to programming. This approach is based on the idea of building applications by discovering and implementing network-available services, or by invoking the available applications to accomplish a task. Web services deliver interoperability, for example, web services applications provide a way for components created in different programming languages to work together as if they were created using the same language. Web services rely on existing transport technologies, such as HTTP, and standard data encoding techniques, such as Extensible Markup Language (XML), for invoking the implementation.

Procedure

1. Identify your goals and design web services to fit your e-business solution. Consider what you want to accomplish by using web services. Decide how web services fit into your current topology, applications and programming model. Determine how the Web services process requests on the server and how the clients manage and use the web service.
2. Design your web services for reliability, availability, manageability and security. For example, you want your web services to process a transaction in a reasonable time at all hours of the day and provide users with optimal security, such as authentication for buyers. Planning to use web services to work with WebSphere Application Server helps to meet these requirements.

3. Review the standards used in developing and deploying web services onto WebSphere Application Server. Development and deployment are based on a variety of Java programming models.
4. Decide what development and implementation tools to use. You can use a variety of manual development and implementation tasks. Whether you have an existing web service to implement or you want to develop your own from a JavaBeans implementation or from an Enterprise JavaBeans (EJB) module, you can choose different tasks respective to your resources. You can also use assembly tools to complete development and implementation tasks.
5. Install the application server. For detailed information on installing the application server, read about installing your application serving environment.
6. Review web services samples.

Results

You have a design plan for implementing web services applications into your business architecture.

Deploying web services

Deploying web services applications onto application servers

After assembling the artifacts required to enable the web module for web services into an enterprise archive (EAR) file, you can deploy the EAR file into the application server.

Before you begin

To deploy Java-based web services, you need an enterprise application, also known as an EAR file that is configured and enabled for web services.

A Java API for XML-Based Web Services (JAX-WS) application does not require additional bindings and deployment descriptors for deployment whereas a Java API for XML-based RPC (JAX-RPC) web services application requires you to add additional bindings and deployment descriptors for application deployment. JAX-WS is much more dynamic, and does not require any of the static data generated by the deployment step required for deploying JAX-RPC applications.

For JAX-WS web services, the use of the `webservices.xml` deployment descriptor is optional because you can use annotations to specify all of the information that is contained within the deployment descriptor file. You can use the deployment descriptor file to augment or override existing JAX-WS annotations. Any information that you define in the `webservices.xml` deployment descriptor overrides any corresponding information that is specified by annotations.

Note: In a mixed node cell, you can only target a JAX-WS enabled enterprise beans module to a server using WebSphere Application Server Version 7.0 and later. However, you can target a JAX-WS enabled web application archives (WAR) module to a server using either WebSphere Application Server Version 7.0 and later or WebSphere Application Server Version 6.1 Feature Pack for Web Services

You can use the wsdeploy command with JAX-RPC applications to add WebSphere product-specific deployment classes to a web services-compatible enterprise application enterprise archive (EAR) file or an application client Java archive (JAR) file.

To install or deploy a JAX-WS application, you only need to install the JAX-WS enabled EAR file. If your web services application contains only JAX-WS endpoints, you do not need to run the wsdeploy command, as this command is used to process only JAX-RPC endpoints.

Ensure that you have installed the HTTP or Java Message Service (JMS) router module that was generated with the endptEnabler command onto the same target as your web services enterprise bean

JAR files. These HTTP or JMS router modules are included in your web services application and they need to use the runtime libraries of the application server.

About this task

This task is one of the steps in developing and implementing web services.

You can use either the administrative console or the wsadmin scripting tool to deploy an EAR file. If you are installing an application containing web services by using the wsadmin command, specify the `-deployws` option for JAX-RPC applications. If you are installing an application containing web services by using the administrative console, select **Deploy WebServices** in the Install New Application wizard. For more information about installing applications using the administrative console, see the installing enterprise application files with the console information.

If your JAX-RPC web services application was previously deployed with the wsdeploy command, it is not necessary to specify web services deployment during installation.

The following actions deploy the EAR file with the wsadmin command:

Procedure

1. Start *install_root*`/bin/wsadmin` from a command prompt.
2. Deploy the EAR file.
   - For JAX-WS web service applications, enter the `$AdminApp install` *EARfile* `"-usedefaultbindings"` command at the wsadmin prompt.
   - For JAX-RPC web service applications, enter the `$AdminApp install` *EARfile* `"-usedefaultbindings -deployws"` command at the wsadmin prompt.

Results

You have a web service installed onto your application server.

Note: While installing web services applications that contain a large number of enterprise beans onto the application server, you might receive out of memory errors. If you receive out of memory errors, increase the heap size of your Java Virtual Machine (JVM). Read about tuning the IBM virtual machine for Java documentation to learn more about tuning the application server environment.

What to do next

You can confirm that the web services application was deployed by entering the web service endpoint URL in a browser, then viewing an informative page. The information page contains the following information:

```
{http://webservice.pli.tc.wssvt.ibm.com}RetireWebServices
Hello! This is an Axis2 web service!
```

The first line of this information is variable, depending on your web service. The URI in the brackets is the namespace and the string that follows, in this example RetireWebServices, is the name of the port used to access the web service.

The next step you might want to consider is to apply security to your web service.

Provide options to perform the web services deployment settings

Use this page to specify options for web services deployment.

This administrative console page is a step in the application installation and update wizards.

To view this page, you must select **Deploy web services** on the **Select installation options** page.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **New application** > ***application_path*** .
2. Select the option to **Show all installation options and parameters** .
3. Click **Next** to get to the **Step: Select installation options** page.
4. Select **Deploy web service**.
5. Click **Next** to get to the **Step: Provide options to perform the web services deployment** page.

You can specify the web services deployment options on this page only when installing or updating an application that uses web services.

The wsdeploy command is supported by Java API for XML-based RPC (JAX-RPC) applications. The Java API for XML-Based Web Services (JAX-WS) programming model that is implemented by the application server does not support the wsdeploy command. If your web services application contains only JAX-WS endpoints, you do not need to run the wsdeploy command, as this command is used to process only JAX-RPC endpoints.

The options that you specify set parameter values for the wsdeploy command. The wsdeploy command adds product-specific deployment classes to a web services-compatible enterprise archive (EAR) file or an application client Java archive (JAR) file. These classes include:

- Stubs
- Serializers and deserializers
- Implementations of service interfaces

The wsdeploy command is run during installation after you click **Finish** on the **Summary** page of the wizard.

Deploy web services option - Classpath:

Specifies entries to add to the CLASSPATH when the generated classes are compiled.

To specify the class paths of multiple entries, you need to separate the entries with a semicolon on Windows platforms and on Linux, Unix, and z/OS platforms, you need to use a colon to separate the entries. This is the same separator that is used with the CLASSPATH environment variable.

This option is the same as the wsdeploy command parameter `-cp` *`class_path`*.

Data type	String
Default	null

Deploy web services option - Extension Directories:

Specifies a directory that contains zipped or Java archive (JAR) files. All zipped and JAR files in this directory are added to the CLASSPATH used to compile the generated files.

This option is the same as the wsdeploy command parameter `-jardir` *`directory`*.

Data type	String
Default	null

wsdeploy command

Use the wsdeploy command to add WebSphere product-specific deployment classes to a web services-compatible enterprise application enterprise archive (EAR) file or an application client Java archive (JAR) file.

The wsdeploy command is supported by Java API for XML-based RPC (JAX-RPC) applications. The Java API for XML-Based Web Services (JAX-WS) programming model that is implemented by the application server does not support the wsdeploy command. If your web services application contains only JAX-WS endpoints, you do not need to run the wsdeploy command, as this command is used to process only JAX-RPC endpoints.

The deployment classes that are added by the wsdeploy tool to a web services-compatible EAR file or a JAR file include:
- Stubs
- Serializers and deserializers
- Implementations of service interfaces

This deployment step must be performed at least once, and can be performed more often. Deployment can be performed separately using the wsdeploy command, assembly tools, or when the application is installed. When using the wsadmin command for installation, specify the **-deployws** option.

The wsdeploy command operates as noted in the following list:
- Each module in the enterprise application or JAR file is examined.
- If the module contains web services implementations, indicated by the presence of the `webservices.xml` deployment descriptor, the associated Web Services Description Language (WSDL) files are located and the WSDL2Java command is run with the role deploy-server option.
- If the module contains web services clients, indicated by the presence of the client deployment descriptor, the associated WSDL files are located and the WSDL2Java command is run with the role deploy-client option.
- The files generated by the WSDL2Java command are compiled and repackaged.

See the WSDL2Java command for JAX-RPC applications command information to learn more about the files that are generated for deployment.

When the generated files are compiled, they can reference application-specific classes outside the EAR or JAR file, if the EAR or JAR file is not self-contained. In this case, use either the -jardir or -cp option to specify additional JAR or zip files to be added to CLASSPATH variable when the generated files are compiled.

wsdeploy command syntax

The command syntax is noted in the following example:

```
wsdeploy Input_filename Output_filename [options]
```

Required options:
- ***Input_filename***

  Specifies the path to the EAR or JAR file to deploy.
- ***Output_filename***

  Specifies the path of the deployed EAR or JAR file. If *output_filename* already exists, it is silently overwritten. The *output_filename* can be the same as the *input_filename*.

Other options:
- **-jardir** *directory*

  Specifies a directory that contains JAR or zip files. All JAR and zip files in this directory are added to the CLASSPATH used to compile the generated files. This option can be specified zero or more times.
- **-cp** *entries*

  Specifies entries to add to the CLASSPATH when the generated classes are compiled. Multiple entries are separated the same as they are in the CLASSPATH environment variable.
- **-codegen**

  Specifies to generate but not compile deployment code. This option implicitly specifies the -keep option.
- **-debug**

Includes debugging information when compiling, that is, use javac -g to compile.

- **-help**

  Displays a help message and exit.
- **-ignoreerrors**

  Do not stop deployment if validation or compilation errors are encountered.
- **-keep**

  Do not delete working directories containing generated classes. A message is displayed indicating the name of the working directory that is retained.
- **-novalidate**

  Do not validate the web services deployment descriptors in the input file.
- **-trace**

  Displays processing information, including the names of the generated files.
- **-compliancelevel** *level*

  Sets the JDK level for compiler compliance. Valid values include: 1.4, 5.0, 6.0 (default) and 7.0. This flag is optional.

Example The following example illustrates how the options are used with the wsdeploy command:

```
wsdeploy x.ear x_deployed.ear -trace -keep
Processing web service module x_client.jar.
Keeping directory: f:\temp\Base53383.tmp for module: x_client.jar.
Parsing XML file:f:\temp\Base53383.tmp\WarDeploy.wsdl
Generating f:\temp\Base53383.tmp\generatedSource\com\test\WarDeploy.java
Generating f:\temp\Base53383.tmp\generatedSource\com\test\WarDeployLocator.java
Generating f:\temp\Base53383.tmp\generatedSource\com\test\HelloWsBindingStub.java
Compiling f:\temp\Base53383.tmp\generatedSource\com\test\WarDeploy.java.
Compiling f:\temp\Base53383.tmp\generatedSource\com\test\WarDeployLocator.java.
Compiling f:\temp\Base53383.tmp\generatedSource\com\test\HelloWsBindingStub.java.
Done processing module x_client.jar.
```

Messages

- Flag *-f* is not valid.

  Option *f* was not recognized as a valid option.
- Flag *-c* is ambiguous.

  Options can be abbreviated, but the abbreviation must be unique. In this case, the wsdeploy command cannot determine which option was intended.
- Flag *-c* is missing parameter *-p*.

  A required parameter for an option is omitted.
- Missing *p* parameter.

  A required option is omitted.

JAX-WS application deployment model

The administration function of the product is enhanced to support installing and deploying Java Application Programming Interface (API) for XML Web Services (JAX-WS) applications like any other WebSphere Application Server applications.

A JAX-WS application is packaged as a web application archive (WAR) file or a WAR module within an Enterprise Archive (EAR) file. The JAX-WS application deployment model is similar to the Java API for XML Remote Protocol Call (JAX-RPC) web services application model. The main difference between them is that JAX-RPC web services application requires you to add additional bindings and deployment descriptors for application deployment. A JAX-WS application does not require additional bindings and deployment descriptors for deployment. You can deploy your JAX-WS applications as you would deploy any other WebSphere Application Server application.

JAX-WS web services is a rewrite of JAX-RPC web services. The table compares the web services stack for both JAX-WS and JAX-RPC web services.

JAX-RPC web services	JAX-WS web services
Bindings are proprietary	Bindings are based on the open source Java API for XML Bindings (JAXB)
Parsing is proprietary	Parsing is based on the open source Java Specification Request (JSR) 173
No Java annotations support	Support for Java annotations such as `@WebService`, `@WebMethod`, `@WebParam`, `@WebResult`, and `@SOAPBinding`
During deployment, some deployment descriptor files are created in a JAX-RPC based service and client. The following files are created on the services side, when it is an EJB based web service and EJB based module: • `webservices.xml` • <name_of_service>`_mapping.xml` • `ibm-webservices-bnd.xmi` • `ibm-webservices-ext.xmi` When the service is a JavaBeans-based or web module-based service, the following files and deployment descriptors are required: • `webservices.xml` • <name_of_service>`_mapping.xml` • In the `web.xml` file, there is no additional content • `ibm-webservices-bnd.xmi` • `ibm-webservices-ext.xmi` The `web.xml` exists in both EJB and JavaBeans based services. However, there is no additional content added to the file during deployment of a Web service application or module.	For JAX-WS web services, the use of the `webservices.xml` deployment descriptor is optional because you can use annotations to specify all of the information that is contained within the deployment descriptor file. You can use the deployment descriptor file to augment or override existing JAX-WS annotations. Any information that you define in the `webservices.xml` deployment descriptor overrides any corresponding information that is specified by annotations.

Note: Starting with WebSphere Application Server Version 7.0 and later, Java EE 5 application modules (web application modules version 2.5 or above, or EJB modules version 3.0 or above) are scanned for annotations to identify JAX-WS services and clients. However, pre-Java EE 5 application modules (web application modules version 2.4 or before, or EJB modules version 2.1 or before) are not scanned for JAX-WS annotations, by default, for performance considerations. In the Version 6.1 Feature Pack for Web Services, the default behavior is to scan pre-Java EE 5 web application modules to identify JAX-WS services and to scan pre-Java EE 5 web application modules and EJB modules for service clients during application installation. Because the default behavior for WebSphere Application Server Version 7.0 and later is to not scan pre-Java EE 5 modules for annotations during application installation or server startup, to preserve backward compatability with the feature pack from previous releases, you must configure either the UseWSFEP61ScanPolicy property in the META-INF/MANIFEST.MF of a web application archive (WAR) file or EJB module or define the Java virtual machine custom property, com.ibm.websphere.webservices.UseWSFEP61ScanPolicy, on servers to request scanning during application installation and server startup. To learn more about annotations scanning, see the JAX-WS annotations information.

Using a third-party JAX-WS web services engine

In certain situations you might need to set up a third-party JAX-WS web services engine. For example, you must set up a third-party JAX-WS web services engine if you need to deploy applications that use a single runtime across various application servers such as WebSphere Application Server, JBoss, and WebLogic, or if you want to build JAX-WS web services applications using third party JAX-WS run-times such as CXF, Axis2, and Metro.

Before you begin

Use of a third-party JAX-WS runtime has limitations. It also requires mandatory configuration changes, and in some cases, it requires manual intervention to resolves issues that occur during deployment and when you run the application. These limitations and issues vary based on the third-party JAX-WS runtime you decide to use. You should understand the limitations for the third-party JAX-WS runtime you are preparing to use before you configure your system to use that implementation.

The following limitations exist regardless of which third-party JAX-WS implementation you use:

- The WebSphere Application Server runtime restricts usage of application modules that use both the JAX-WS implementation provided with WebSphere Application Server, and an external JAX-WS implementation in the same application EAR file. You must use either the JAX-WS implementation provided with WebSphere Application Server or the external implementation in a single application EAR file. This limitation ensures that the runtime WebSphere Application Server does not conflict with the external third-party JAX-WS implementation.
- You must remove any conflicting JAR files from your application library before you deploy an application that uses an external JAX-WS implementation. Most of the external third-party JAX-WS runtimes include some JAR file libraries that are already installed on WebSphere Application Server. This situation causes conflicts in your application library.
- After an application that uses a third-party JAX-WS runtime is deployed on WebSphere Application Server, it is not recognized as a service client or provider. Therefore, you cannot attach application level policy sets to these applications. You must rely on external runtimes support quality of service. Following is a list of WebSphere Application Server features that are not available if you decide to deploy and run application that uses third-party JAX-WS implementations:
  - WS-Security, WS-RM, and WS-Transactions policy sets
  - WSDM
  - JNDI lookup to retrieve JAX-WS Service or Port Instance.

gotcha: Even though IBM supports the enablement of third party JAX-WS runtimes to run on WebSphere Application Server, and ensures the successful deployment of applications that use such runtimes, IBM does not provide support for resolving JAR file conflict problems, or any problem that a stack trace indicates is in the third party code.

About this task

When you deploy an application EAR file with a third-party JAX-WS implementation on WebSphere Application Server, the WebSphere Application Server runtime must ensure the use of the third-party engine, and disable the use of the existing WebSphere Application Server JAX-WS web services engine.

WebSphere Application Server does not claim support for any of the third-party JAX-WS runtimes, but has tested the deployment and execution of applications that use such runtimes.

You must complete the following steps before you can use an external JAX-WS runtime in an application.

Procedure

1. Set the class loader policy to `Classes loaded with local class loader first (parent last)` at the module level.

   Changing the class loader policy to parent last ensures that the external third-party JAX-WS runtime and their dependent library JAR files are first in the class loader search path, thereby ensuring that the third-party implementation is used instead of the WebSphere Application Server.

   a. In the administrative console, click **Applications > Application Types > WebSphere enterprise applications >** *application_name* **> Class loading and update detection**.

   b. Under Class reloading options, select **Override class reloading settings for web and EJB modules** .

c. Under Class loader order, select **Class loader order** property to `Classes loaded with local class loader first (parent last)`.

a. Click **OK**, and then **Save** to save your changes.

2. Turn off web services annotation scanning.

   Annotation scanning can be turned off at the application level or at the server level.

   To turn off annotation scanning at the application level, set the DisableIBMJAXWSEngine property in the META-INF/MANIFEST.MF of a WAR file or EJB module to true. Example:

```
Manifest-Version: 1.0
DisableIBMJAXWSEngine: true
```

   To turn off web services annotation scanning at the server level:

   a. In the administrative console, go to the Custom properties page for the Java virtual machine.

      Servers > Server Types > WebSphere application servers > *server_name*, and then, under Server Infrastructure, click **Java and process management > Process definition > Java virtual machine > Custom properties**

   b. Set the **com.ibm.websphere.webservices.DisableIBMJAXWSEngine** property to `true`

      If this property does not already exist for your configuration, click **New**, and add `com.ibm.websphere.webservices.DisableIBMJAXWSEngine` in the **Name** field and `true` in the **Value** field.

Results

What to do next

- Deploy and run an application EAR file with a third-party JAX-WS implementation on WebSphere Application Server.

Deploying web services client applications

After you have created an enterprise archive (EAR) file for the web services client application, you can deploy the web services client application into the Application Server.

Before you begin

To deploy a Java-based web services client, you need an enterprise application, also known as an enterprise archive (EAR) file that is configured and enabled for web services.

A Java API for XML-Based Web Services (JAX-WS) application is packaged as a web application archive (WAR) file or a WAR module within an Enterprise Archive (EAR) file. A JAX-WS application does not require additional bindings and deployment descriptors for deployment whereas a Java API for XML-based RPC (JAX-RPC) web services application requires you to add additional bindings and deployment descriptors for application deployment. JAX-WS is much more dynamic, and does not require any of the static data generated by the deployment step required for deploying JAX-RPC applications. For JAX-RPC web services clients, you must configure the client deployment descriptors.

About this task

You can use either the administrative console or the wsadmin scripting tool to deploy an EAR file. If you are installing an application containing web services by using the wsadmin command, specify the `-deployws` option for JAX-RPC applications.

Use the wsdeploy command only with JAX-RPC applications. The wsdeploy command is not applicable for JAX-WS applications.

If you are installing an application containing web services by using the administrative console, select **Deploy WebServices** in the Install New Application wizard. Read about installing a new application for more information on using the administrative console.

The following actions deploy the EAR file with the wsadmin command:

Procedure

1. Start *install_root*/bin/wsadmin from a command prompt.
2. Deploy the EAR file.
   - For JAX-WS web service applications, enter the `$AdminApp install` *EARfile* `"-usedefaultbindings"` command at the wsadmin prompt.
   - For JAX-RPC web service applications, enter the `$AdminApp install` *EARfile* `"-usedefaultbindings -deployws"` command at the wsadmin prompt.

Results

You have a deployed a web service client in the application server runtime environment.

What to do next

Test the web services client. You can now test a web services-enabled managed client EAR file or an unmanaged client JAR file.

Making deployed web services applications available to clients

You can publish WSDL files to the file system. If you are a client developer or a system administrator, you can use WSDL files to enable clients to connect to web services.

Before you begin

The publish WSDL administrative console panel supports both JAX-RPC and JAX-WS services. The publish WSDL panel generates a compression file that contains WSDL files for all modules in an application that contains JAX-WS or JAX-RPC web services. Read about providing the HTTP endpoint URL information to learn how the URL information affects the content of the published WSDL.

To publish a Web Services Description Language (WSDL) file you need an enterprise application, also known as an enterprise archive (EAR) file, that contains a Web services-enabled module and has been deployed into WebSphere Application Server. To learn how to deploy web services, see the deploying web services applications onto application servers information.

About this task

The purpose of publishing the WSDL file is to provide clients with a description of the web service, including the URL identifying the location of the service.

After installing a web services application, and optionally modifying the endpoint information, you might need WSDL files containing the updated endpoint informations to make deployed web services application to be available to clients.

Before you publish a WSDL file, you can configure web services to specify endpoint information in the form of URL fragments to enable full URL specification of WSDL ports. Refer to the tasks describing configuring endpoint URL information.

The WSDL files for each web services-enabled module are published to the file system location you specify. You can provide these WSDL files to clients that want to invoke your Web services.

You can specify endpoint information for HTTP ports, for Java Message Service (JMS) ports, or you can directly access enterprise beans that are acting as web services.

Procedure

1. Configure the web services client bindings.
2. Configure the URL endpoint information for HTTP bindings. Do one of the following depending on what kind of bindings you are using:
   - Configure the URL endpoint information for JMS bindings.
3. Externalize or publish the WSDL file out of the application. You can complete this task in the following ways:
   - Publish a WSDL file with the wsadmin tool.

What to do next

Apply security to your web services. To learn more, see the securing web services applications using message level security information.

Configuring web services client bindings

When a web services application is deployed into WebSphere Application Server, an instance is created for each application or module. The instance contains deployment information for the web module or Enterprise JavaBeans (EJB) module, including client bindings.

Before you begin

Deploy a web service into your WebSphere Application Server instance. Read about deploying web services applications onto application servers.

You must know the topology of the URL endpoint address of the web services servers and which web service the client depends upon. You can view the deployment descriptors in the administrative console to find the topology information. To learn more, see the View web services server deployment descriptors information.

About this task

The client bindings define the Web Services Description Language (WSDL) file name and preferred ports. The relative path of a web service in a module is specified within a compatible WSDL file that contains the actual URL to be used for requests. The address is only needed if the original WSDL file did not contain a URL, or when a different address is needed. For a service endpoint with multiple ports, you need to define an alternative WSDL file name.

The following steps describe how to edit bindings for a web service after these bindings are deployed on a server. When one web service communicates with another web service, you must configure the client bindings to access the downstream web service.

To configure client bindings through the administrative console:

Procedure

1. Open the administrative console.
2. Click **Applications** > **Enterprise Applications** > *application_instance* > **Manage Modules** > *module_instance* > **Web services client bindings**.
3. Find the web service you want to update.

   The web services are listed in the **Web Service** field.
4. Select the WSDL file name from the drop down box in the WSDL file name field.

5. Click **Edit** in the Preferred port mappings field to configure the default port to use.
   a. Specify the port type and the preferred ports in the Port type and Preferred ports fields.

      Configuring the preferred port enables you to select an optimal port implementation use non-SOAP protocols. See the RMI-IIOP web services using JAX-RPC information to learn more about using non-SOAP protocols.
   b. Click **Apply** and **OK**.
6. Click **Edit** in the Port information field to configure the request timeout, the overridden endpoint, and the overridden binding namespace for a port.

   Configuring the request timeout accommodates complex topologies that can have multiple cascaded Web services that involve multiple hops or long-running services.

   You can configure Timeout values based on observed behavior of the overall system as integration proceeds. For example, a web service client might time out because of changing network conditions or the performance of an external web service. When you have applications containing web services clients that timeout, you can change the request time out values for the clients.

   You can specify an endpoint URL to override the current endpoint. A client invoking a request on this port uses this endpoint instead of the endpoint specified in the WSDL file. You can specify the **Overridden endpoint URL** value for both Java API for XML-Based Web Services (JAX-WS) clients and Java API for XML-based RPC (JAX-RPC) clients.

   Note: The **Overridden endpoint URL** field is applicable for both JAX-WS and JAX-RPC clients. The other fields on this administrative console page are only applicable for JAX-RPC clients.
   a. Click **Apply** and **OK**.

Results

Your web service client bindings are configured.

What to do next

Now you can finish any other configurations, start or restart the application, and verify the expected behavior of the web service.

Web services client bindings:

The client bindings define the Web Services Description Language (WSDL) file name, preferred ports and other port information. Use this page to specify the client bindings and the port mappings for the web services in a module.

A web service can specify the relative path within the module of a compatible WSDL file containing the actual URL to be used for requests. The relative path only needs to be specified if the original WSDL file does not contain a URL or when a different URL is needed. For a service endpoint with multiple ports defined, a preferred mapping specifies the default port to use for a port type.

To view this administrative console page, complete the following steps:
1. Click **Applications > Application Types > WebSphere enterprise applications >** ***application_name*** .
2. Click **Manage modules >** ***URI_file_name*** **> web services client bindings** .

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

Web service:

Identifies the name of this web service. A module can contain one or more web services.

EJB:

Identifies the name of the EJB for the EJB modules.

WSDL file name:

Specifies the WSDL file name, which is relative to the module. Locate the WSDL file name in the drop down menu.

Preferred port mappings:

Specifies and manages the preferred port type mapping for a web service when a particular port type is requested.

Click **Edit** to edit the preferred port mapping information on the **Preferred port mappings** panel.

Port information:

Specifies additional configuration information for the ports of this web service.

Click **Edit** to edit the port information on the **Port information** panel. You can set a request timeout, override an endpoint and override a binding namespace for each client port.

Preferred port mappings:

Use this page to view and manage a preferred portType mapping for a web service.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

When you have multiple ports that reference the same portType (service endpoint interface), a preferred port specifies the port to use when the `Service.getPort(Class SEI)` method is called with only the service endpoint interface.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Manage Modules** > ***module instance*** > **Web services client bindings** > **Edit** > ***preferred_port_instance***.

portType:

Specifies the portType.

The preferred port and the portType values are both of the type `java.xml.namespace.QName`.

Preferred port:

Specifies the preferred port to be associated with a particular portType. The `Service.getPort(Class)` method returns the preferred port associated with the specified service endpoint interface class (portType).

The preferred ports available are listed, as well as a value of `None`, which indicates no preferred port is selected.

Web services client port information:

Use this page to specify a request timeout, override an endpoint, and override a binding namespace for a web services client port.

A web service can have multiple ports. You can view and configure the port attributes for each defined web service port. The web services are listed on the web services client bindings panel.

To view this page, click **Applications** > **Application Types** > **WebSphere enterprise applications** ***application_name*** > **Manage Modules** > ***module_instance*** > **Web services client bindings** > **Edit** .

This administrative console panel applies to both Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services. The **Overridden endpoint URL** field is the only field supported for JAX-WS clients. The other fields are not applicable for JAX-WS clients.

Port:

Specifies the name of a port.

Request timeout:

Specifies the time, in seconds, that a web service client waits for a request to complete on this port. If a timeout is not specified, the default request timeout for the client to wait is `300` seconds. If the value is set at `0` (zero), the timeout used is the default value for the underlying transport mechanism. This field is supported only for JAX-RPC clients.

A typical use for this setting is to customize the client's behavior when it is configured to use a JMS transport to access a web service to make it wait longer for an expected completion. Depending upon network conditions, or the nature of a web service implementation, it might be necessary to tune the timeout.

Overridden endpoint URL:

Specifies the name of an endpoint that is used to override the current endpoint. A client invoking a request on this port uses this endpoint instead of the endpoint specified in the WSDL file. This field is supported for both JAX-WS and JAX-RPC clients.

If an assembled application contains a web service client that is statically bound, the client is locked into using the implementation (service end point) identified in the WSDL file used during development. Overriding the endpoint is an alternative to configuring the deployed WSDL attribute.

The overridden endpoint URI attribute is specified on a per port basis. It does not require an alternative WSDL file within the module. The overridden endpoint URI takes precedence over the deployed WSDL attribute. The client uses this value for the service end point URI or SOAP address, instead of the value in the static client bindings.

Note: You can edit this field if you have managed clients or a mixture of both managed and unmanaged clients. You cannot edit the field if you have unmanaged clients only.

If you do not want a request by an unmanaged JAX-WS client service to be sent to the endpoint URL that is specified in this field, you can specify the `com.ibm.ws.websvcs.unmanaged.client.dontUseOverriddenEndpointUri` Java virtual machine (JVM) system property. For more information about this custom property, read about the Java virtual machine custom properties.

Overridden binding:

Specifies the WSDL file binding namespace URI to use with this port, instead of the namespace in the WSDL file. This binding does not need to exist in the WSDL file. A client invoking a request on this port

uses this binding instead of the binding specified in the WSDL file. An overridden binding namespace cannot be specified unless an overridden endpoint is specified. This field is supported only for JAX-RPC clients.

Configuring endpoint URL information for HTTP bindings

Configuring a service endpoint is necessary to connect Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services clients to any web services among the components being assembled or to any external web services.

Before you begin

You can develop an HTTP accessible Java API for XML-based remote procedure call (JAX-RPC) or Java API for XML Web Services (JAX-WS) web service when you have an existing JavaBeans object to enable as a web service. For additional information, see the using HTTP to transport web services requests information.

You can use either the administrative console or property files to configure and manage HTTP endpoint URL fragments. To learn about using property files to set and manage the URL fragments, see the information about working with web services endpoint URL fragment property files.

This task describes using the administrative console to configure endpoint URL information for HTTP bindings.

About this task

You can specify HTTP URL prefixes for web services that are accessed through HTTP by using the Provide HTTP endpoint URL information panel in the administrative console. The HTTP URL prefixes provide location specific information and are used to form complete endpoint URLs that are included within published WSDL files.

Note: The Provide HTTP panel in the administrative console displays modules that contain Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services. You can use the Provide HTTP panel to provide URL information for both types of web services, however, the panel does not indicate which type of service that you are working with.

To configure these prefixes with the administrative console:

Procedure

1. Open the administrative console.
2. Click **Applications** > **Enterprise Applications** > *application_instance* > **Provide HTTP endpoint URL information**.
3. Specify the URL prefixes for the web service.

   In this step you specify the protocol (HTTP or HTTPS), as well as the *host_name* and *port_number* used in the endpoint URL. You can select a prefix from a predefined list, by selecting the default HTTP URL prefix, or you can use a custom HTTP URL prefix.

   a. Select **Default HTTP URL prefix** or **Custom HTTP URL Prefix**.

      If you select the default HTTP URL prefix, a list provides you with a choice of endpoint URL prefixes. The list is a combination of two sets of ports in the module: the virtual host ports and the application server ports. Use a prefix from this list if the application server of the web service is accessed directly. Select a value and also select the check box of the modules to use the prefix.

      If you want to use a custom HTTP URL prefix, type the value in the field. Select the check box to use in the prefix.

      If you configure a custom HTTP URL prefix, , you must also configure the custom JVM property, `com.ibm.ws.webservices.enableHTTPPrefix` in the administrative console and set the value to `true`.

You must restart the application server after this custom property has been defined so that this property is used by the system. Setting this custom JVM property is required so the custom HTTP endpoint prefix information is correctly displayed in the ?WSDL query that is returned from the browser and the URL field of the WSDL file that is returned to the client. If this custom property is not defined with the value of `true`, the custom HTTP URL prefix is not reflected in the WSDL file that the service returns to the client. To learn how to configure this custom JVM property, see the documentation on configuring additional HTTP transport properties using the JVM custom property panel in the administrative console.

Note: The `com.ibm.ws.webservices.enableHTTPPrefix` custom property applies to JAX-RPC web services applications only.

b. Click **Apply**.

The URL prefix, whether default or custom, is copied to the selected module HTTP URL prefix field.

c. Click **OK**. The URL information is saved to your workspace.

Results

You have specified the partial URL information that is used to form the target endpoint addresses in the WSDL files that are published using the Publish WSDL files panel.

What to do next

Configure any other URL endpoint information for Java Message Service (JMS) bindings and direct Enterprise JavaBeans (EJB) access. Then publish the WSDL files to make the deployed web services application available to clients.

Provide HTTP endpoint URL information:

Use this page to specify endpoint URL prefix information for web services accessed by HTTP. Prefixes are used to form complete endpoint addresses included in published Web Services Description Language (WSDL) files.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Provide HTTP endpoint URL information**.

You can specify a portion of the endpoint URL to be used in each web service module. In a published WSDL file, the URL defining the target endpoint address is found in the location attribute of the port's `soap:address` element.

This administrative console page applies for Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services.

In addition to the using the administrative console, you can use property files to configure and manage HTTP endpoint URL fragments. To learn about using property files to set and manage the URL fragments, see the information about working with web services endpoint URL fragment property files.

Specify endpoint URL prefixes for web services:

Specifies the *protocol* (either http or https), *host_name*, and *port_number* to be used in the endpoint URL.

You can select a prefix from a predefined list using the **HTTP URL prefix** or **Custom HTTP URL prefix** field.

The URL prefix format is *protocol://host_name:port_number*, for example, `http://myHost:9045`. The actual endpoint URL that is contained in a published WSDL file consists of the prefix followed by the module's context-root and the web service url-pattern, for example, `http://myHost:9045/services/myService`.

Select default HTTP URL prefix:

Specifies the drop down list associated with a default list of URL prefixes. This list is the intersection of the set of ports for the module's virtual host and the set of ports for the module's application server. Use items from this list if the web services application server is accessed directly.

To set an HTTP endpoint URL prefix, select **Select default HTTP URL prefix** and select a value from the drop down list. Select the check box of the modules that are to use the prefix and click **Apply**. When you click **Apply**, the entry in the **Select default HTTP URL prefix** or **Select custom HTTP URL prefix** fields, depending on which is selected, is copied into the **HTTP URL prefix** field of any module whose check box is selected.

Select custom HTTP URL prefix:

Specifies the *protocol*, *host*, and *port_number* of the intermediate service if the web services in a module are accessed through an intermediate node, for example the web services gateway or an IHS server.

To set a custom HTTP endpoint URL prefix, you must also configure the custom JVM property, `com.ibm.ws.webservices.enableHTTPPrefix` in the administrative console and set the value to `true`. Setting this custom JVM property is required so the custom HTTP URL is correctly populated in the URL field of the WSDL file that is returned to the client. If this custom JVM property is not configured, the custom HTTP URL prefix is not in the URL field in the copy of the WSDL file that the service returns to the client. To learn how to configure this custom JVM property, see the documentation on configuring additional HTTP transport properties using the JVM custom property panel in the administrative console. You must restart the application server after this custom property has been defined so that this property is used by the system.

After the `com.ibm.ws.webservices.enableHTTPPrefix` custom JVM property is configured, select **Select custom HTTP URL prefix** and enter a value. Select the check box of the modules that are to use the prefix and click **Apply**. When you click **Apply**, the entry in the **Select default HTTP URL prefix** or **Select custom HTTP URL prefix** fields, depending on which is selected, is copied into the **HTTP endpoint URL prefix** field of any module whose check box is selected.

Note: The `com.ibm.ws.webservices.enableHTTPPrefix` custom property applies to JAX-RPC web services applications only.

Configuring endpoint URL information for JMS bindings

WebSphere Application Server supports the use of the Java Message Service (JMS) API to transport web services requests, as an alternative to using HTTP.

Before you begin

The application server supports use of the Java Message Service (JMS) API to transport web services requests, as an alternative to HTTP transport. Read about using the Java Message Service (JMS) to transport web services requests to learn more about how web service clients and servers can communicate through JMS queues and topics instead of through HTTP connections.

You can use either the administrative console or property files to configure and manage JMS endpoint URL fragments. To learn about using property files to set and manage the URL fragments, see the information about working with web services endpoint URL fragment property files.

This task describes using the administrative console to configure endpoint URL information for JMS bindings.

About this task

Configuring a service endpoint is necessary to connect web service clients to any web services among the components being assembled or to any external web services. You can configure the endpoint URL information for JMS during application installation

In this task, enter the JMS endpoint URL prefix to use for each web service-enabled Enterprise JavaBeans (EJB) Java archive (JAR) file that belong to the application. The JMS endpoint URLs are included in the Web Services Description Language (WSDL) files published for clients to use.

You can specify HTTP URL prefixes for web services that are accessed through HTTP by using the Provide HTTP endpoint URL information panel in the administrative console. These prefixes are used to form complete endpoint addresses that are included in WSDL files when published.

You can specify JMS URL prefixes by using the Provide JMS and EJB endpoint URL information panel in the administrative console during or after application installation.

This task applies for Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services.

To configure JMS URL prefixes:

Procedure

1. Open the administrative console.
2. Click **Applications** > **Enterprise Applications** > ***application_instance*** > **Provide JMS and EJB endpoint URL information**.
3. Locate the list of web services modules that are accessible through JMS transport.
4. Type the JMS URL fragment in the **URL fragment** field. Enter a URL fragment that is a prefix to the initial URL part that is obtained by examining the deployment information of the Web service. See the usage scenario following this task for more information.

   The value that you enter is used to define the location attribute of the port soap:address element within the WSDL file that is published using the *application_name*_ExtendedWSDLFiles.zip or the *application_name*_WSDLFiles.zip file on the Publish WSDL zip files panel.

Results

You have a web service that is accessible through the JMS transport and configured with JMS bindings.

Example

Suppose an application called StockQuoteService contains an EJB JAR file that is named StockQuoteEJB, which contains one or more web services that are accessible through the JMS transport.

See the using SOAP over Java Message Service to transport web services information to review the example that defines a queue with the Java Naming and Directory Interface (JNDI) name of jms/StockQuote_Q, and a connection factory with the JNDI name of jms/StockQuote_CF, for your application.

In this example, you specify the following string as the JMS URL prefix within the Provide JMS and EJB endpoint URL information panel:

```
jms:/queue?destination=jms/StockQuote_Q&connectionFactory=jms/StockQuote_CF
```

The WSDL publisher uses this partial URL string to produce the actual JMS URL for each port component that is defined in the module. The `targetService=<port_name>` string is added to the end of the JMS URL, for example:

```
jms:/queue?destination=jms/StockQuote_Q&connectionFactory=jms/StockQuote_CF&targetService=getQuote
```

The published WSDL file is used by clients to invoke the web service.

What to do next

Publish the WSDL files to make the deployed web services application available to clients.

Provide JMS and EJB endpoint URL information:

Use this page to specify Java Message Service (JMS) and Enterprise JavaBeans (EJB) endpoint URL fragments for web services accessed through SOAP and Java Message Service (JMS) or directly as enterprise beans. Fragments are used to form complete endpoint addresses included in published Web Services Description Language (WSDL) files.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Provide JMS and EJB endpoint URL information**.

You can specify a fragment of the endpoint URL to be used in each web service module. In a published WSDL file, the URL defining the target endpoint address is found in the location attribute of the port's `soap:address` element.

If you are using web services modules that are configured to use JMS or configured to access enterprise beans directly, these modules are listed on this panel.

This administrative console page applies for Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services.

In addition to the using the administrative console, you can use property files to configure and manage JMS and EJB endpoint URL fragments. To learn about using property files to set and manage the URL fragments, see the information about working with web services endpoint URL fragment property files.

URL fragment for JMS:

Specifies a URL fragment for web services accessed through a JMS transport. You can enter a value that is used to define the `soap:address` of a web service. When WSDL files are published, a URL is formed using this fragment and is contained in the WSDL files.

The URL fragment that is entered as a value is a prefix to which the targetService property is appended to form a complete JMS URL endpoint. The default value is obtained by examining the installed service's deployment information, for example, `jms:jndi:jms/MyQueue&jndiConnectionFactoryName=jms/MyCF`.

This information is obtained from the configured JMS endpoint for the web service, which is a Message Driven Bean (MDB) defined by the **endpointEnabler** command-line tool. You can modify the URL fragment, for example, by adding properties. The URL fragment is combined with the targetService property to form the complete URL, for example, `jms:jndi:jms/MyQueue&jndiConnectionFactoryName=jms/MyCF&priority=5&targetService=GetQuote`.

URL fragment for EJB:

Specifies a URL fragment for web services accessed through an EJB binding. You can enter a value used to define the location attribute of the port's `generic:address` element of a Web service. This port address

is contained in the WSDL compression file when the compression file is published using the *application_name_ExtendedWSDLFiles*.zip field on the **Publish WSDL zip file** panel.

The URL fragment value entered is a suffix, which is appended to the initial part of the URL obtained by examining the web service's deployment information. For example, the following URL fragment can be obtained from the EJB's deployment information: wsejb:/com.acme.sample.MyStockQuoteHome?jndiName=ejb/MyStockQuoteHome.

In this case, you can enter the following information in the URL fragment field, jndiProviderURL=corbaloc:iiop:myhost.mycompany.com:2809, which results in this endpoint URL, wsejb:/com.acme.sample.MyStockQuoteHome?jndiName=ejb/MyStockQuoteHome&jndiProviderURL=corbaloc:iiop:myhost.mycompany.com:2809.

Configuring endpoint URL information to directly access enterprise beans

WebSphere Application Server supports directly accessing an enterprise bean as a web service, as an alternative to using HTTP or Java Message Service (JMS) to transport requests between the server and the client. The Enterprise JavaBeans (EJB) module that is used as a web service contains a Web Services Description Language (WSDL) file that contains EJB bindings.

Before you begin

To learn more about the process of directly accessing an enterprise bean as a web service, see the using WSDL EJB bindings to invoke an EJB from a JAX-RPC web services client.

You can use either the administrative console or property files to configure and manage EJB endpoint URL fragments. To learn about using property files to set and manage the URL fragments, see the information about working with web services endpoint URL fragment property files.

This task describes using the administrative console to configure endpoint URL information to directly access enterprise beans.

About this task

Configuring a service endpoint is necessary to connect web service clients to any web services among the components being assembled or to any external web services.

You can specify web address endpoints of the enterprise bean for web services that are accessed directly by EJB bindings using the Provide JMS and EJB endpoint web address information panel in the administrative console.

If you have modules that are configured for using direct EJB access, the modules are listed on the Provide JMS and EJB endpoint web address information panel in the administrative console. The EJB endpoint is only available in the WSDL that is found in the *application_name*_ExtendedWSDLfiles.zip file.

You can specify a fragment of the endpoint web address for the web services in each module.

To configure the web address endpoints of the enterprise bean with the administrative console:

Procedure

1. Open the administrative console.
2. Click **Applications** > **Enterprise Applications** > ***application_instance*** > **Provide JMS and EJB endpoint URL information**.
3. Locate the list of EJB modules.
4. Select the application module.
5. Type the web address fragment in the **URL fragment** field.

Enter a web address fragment that is a suffix to the initial web address part that is obtained by examining the web service deployment information. See the example following this task for more information.

The value that you enter is used to define the location attribute of the port generic:address element within the WSDL file that is published using the *application_name*_ExtendedWSDLFiles.zip file name link on the Publish WSDL zip files panel. The zip file names are listed as links on the panel.

6. Click **OK**.
7. Click **Save**.

Results

You have configured endpoints of the enterprise bean for Web services that are accessed directly by EJB bindings.

Example

The following example illustrates a web address fragment to enter in the URL fragment field.

The following web address information can be obtained from the deployment descriptor of an enterprise bean:

```
wsejb:/com.acme.sample.MyStockQuoteHome?jndiName=ejb/MyStockQuoteHome
```

Enter the following web address fragment in the URL fragment field:

```
jndiProviderURL=corbaloc:iiop:myhost.mycompany.com:2089
```

The results are shown in the following example:

```
wsejb:/com.acme.sample.MyStockQuoteHome?jndiName=ejb/MyStockQuoteHome&jndiProviderURL=
corbaloc:iiop:myhost.mycompany.com:2089
```

What to do next

Provide a description of the web service to the service requestor by publishing WSDL files. To learn more, read about making deployed web services applications available to clients.

Publishing WSDL files using the administrative console

You can publish a Web Services Description Language (WSDL) file using the WebSphere Application Server administrative console.

Before you begin

Before completing this task, you need to install or deploy the web service. After deployment, configure the URL endpoint tasks for your transport:

- Configure endpoint URL information for HTTP bindings
- Configure endpoint URL information for JMS bindings
- Configure endpoint URL information to directly access enterprise beans

About this task

By publishing a WSDL file, you are providing clients with a description of the web service, including the URL identifying the location of the service.

The WSDL files in each web services-enabled module are published to the file system location you specify. You can provide these WSDL files in the development and configuration process of the web service clients so they can invoke your web services.

This task applies for Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services.

To learn about more ways to publish WSDL files, see the making deployed web services applications available to clients information.

To publish an application's WSDL file with the administrative console:

Procedure

1. Open the administrative console.
2. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
3. Under Web Services Properties, click **Publish WSDL files**. This takes you to the **Publish WSDL zip files** page.
4. Click the WSDL compression file to download. The compression file contains the application's published WSDL files. The compression file `ExtendedWDLFiles.zip` contains EJB binding information. It can also contain JMS or HTTP binding information. The compression file `WSDLFiles.zip` only contains JMS or HTTP binding information.

What to do next

Apply security to your web services. To learn more, see the securing web services applications using message level security information.

Publish WSDL compressed files settings:

Use this page to publish Web Services Description Language (WSDL) files.

This administrative console page applies for Java API for XML-Based Web Services (JAX-WS) and Java API for XML-based RPC (JAX-RPC) web services.

The publish WSDL panel generates a compressed file that contains WSDL files for all modules in an application that contains a JAX-WS or JAX-RPC web service. Read about providing the HTTP endpoint URL information to learn how the URL information affects the content of the published WSDL.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **Publish WSDL files**.

When you click **OK**, a panel showing one or several compressed file names displays. Each compressed file contains a WSDL file that represents the web services-enabled modules in the application. When you select a compressed file to publish, a dialogue displays from which you can choose where to create the compressed file. Within the published compressed files, the directory structure is *`application_name`*`/`*`module_name`*`/[META-INF|WEB-INF]/wsdl/`*`wsdl_file_name`*.

In a published WSDL file, the location attribute of a port's `soap:address` element contains the endpoint URL through which the web service is accessed. Using the **Provide HTTP endpoint URL information** and the **Provide JMS and EJB endpoint URL information** panels, configure the endpoint URLs to be used for the web services in each module.

application_name_WSDLFiles.zip:

Specifies the *`application_name`*`_WSDLFiles.zip` file containing the WSDL that describes web services that are accessible by standard SOAP-based ports.

application_name_ExtendedWSDLFiles.zip:

Specifies the *application_name*_ExtendedWSDLFiles.zip file containing the WSDL file that describes the web services available, including SOAP-based and non-SOAP based (for example, EJB) ports.

If there are no web services configured for direct EJB access, this compressed file name is not displayed. Do not use this compressed file if you want to produce a WSDL file compliant to standards.

Publishing WSDL files using a URL

You can publish a Web Services Description Language (WSDL) file using a URL.

Before you begin

Before you can publish a WSDL file using a URL, ensure the web services-enabled application is installed and running.

The files referenced by the <wsdl-file> element in the webservices.xml might import other WSDL or XML Schema Definition (XSD) files. Typically, all WSDL or XSD files are initially placed into the META-INF/wsdl directory when using Enterprise JavaBeans (EJB) or the WEB-INF/wsdl directory when using JavaBeans. If your WSDL or XSD files are not placed in one of these directories, the file referenced by the <wsdl-file> and its imported files are copied to the wsdl directory for publishing purposes.

There are two different forms of URL query strings. The first appends /wsdl to the service and returns only HTTP and JMS bindings. The second appends /extwsdl to the service and returns the extended WSDL file, including HTTP, JMS, and EJB bindings. If a WSDL file contains only EJB bindings and the /wsdl query is used, an error message displays in the browser saying there are no HTTP or JMS bindings in the WSDL file. The error message suggests using the /extwsdl query instead. Publishing a WSDL file using a URL requires that the application have a web module; either provided by the application or in the form of an HTTP router module. If an EJB application contains a WSDL file with only JMS or EJB Web service bindings, the endptEnabler command can be used to add an HTTP router module to the application.

Note: Only HTTP URLs are supported for publishing.

About this task

To publish a WSDL file using a URL:

Procedure

1. Retrieve the outer-most WSDL file. The outer-most WSDL file is the WSDL file defined by the <wsdl-file> element in the webservices.xml file.

   Each web service has an endpoint address, like http://example.com/services/stockquote. You can retrieve the outer-most WSDL file (defined by the <wsdl-file> element within the webservices.xml file) by appending the string "/wsdl" or "/wsdl/" to the endpoint address, for example, http://example.com/services/stockquote/wsdl.
2. Retrieve the imported WSDL files. When the outer-most WSDL file imports other WSDL or XSD files, these imported files can be retrieved by appending the relative path to the URL, which is used to retrieve the outer-most WSDL file. This is also true for WSDL files that import other files. This process is similar to the use of relative hyperlinks in HTML documents. If an HTML document contains a hyperlink to other documents, the relative path is appended to create the URL to access the hyperlinked documents.

Example

Suppose you have an application with the following directory structure:

```
<module-root>/
 WEB-INF/
  webservices.xml    /* the <wsdl-file> element points to "WEB-INF/wsdl/fooImpl.wsdl"*/
  web.xml
```

```
ibm-webservices-bnd.xml

wsdl/
 fooImpl.wsdl   /* imports foo.wsdl which is an interface wsdl */
 foo.wsdl       /* type definition for the interface */
```

If the SOAP address for the foo service is `http://examples.com:9080/services/foo`, the simple way to retrieve the foo service's outer-most WSDL is with the following form: `http://examples.com:9090/services/foo/wsdl` or `http://examples.com:9090/services/foo/wsdl/`. The URL is redirected to `http://examples.com:9090/services/foo/wsdl/fooImpl.wsdl`, where `fooImpl.wsdl` is the name of the outer-most WSDL file.

Since the `fooImpl.wsdl` file has the import `<import namespace="http://examples.com/foo" location="a/b/foo.wsdl>`, use the URL `http://examples.com:9090/services/foo/wsdl/a/b/foo.wsdl` to obtain the `foo.wsdl` file.

Running an unmanaged web services JAX-RPC client

WebSphere Application Server Version 8.0 and the Application Client for WebSphere Application Server Version 8.0 provides a thin Java Platform, Standard Edition 6 (Java SE 6) web services client runtime implementation that is based on the Java API for XML-based RPC (JAX-RPC) 1.1 specification. The Thin Client for JAX-RPC with WebSphere Application Server is a stand-alone Java SE 6 client environment that enables running unmanaged JAX-RPC web services client applications in a non-WebSphere environment to invoke web services that are hosted by the application server.

Before you begin

Note: You can use the Thin Client for JAX-RPC with WebSphere Application Server as a stand-alone client run time in a pure Java SE environment, or within an OSGi environment. The Thin Client for JAX-RPC is not supported when running within WebSphere Application Server or WebSphere Application Client environments. In this version of the application server, with the exception of the Administration Thin Client, other Thin Client run times provided with the application server can also reside in the CLASSPATH and coexist with the Thin Client for JAX-RPC.

Before you can set up a JAX-RPC unmanaged client environment you will need to obtain the Thin Client for JAX-RPC Java archive (JAR) file. To obtain the Thin Client for JAX-RPC, you must either install the application server or the application client.

The Thin Client for JAX-RPC JAR file, `com.ibm.ws.webservices.thinclient_8.0.0.jar`, is located in the *app_server_root*`\runtimes` directory. Refer to the license agreements to ensure correct usage and for limitations on copies of the Thin Client for JAX-RPC outside of the WebSphere environment.

The Thin Client for JAX-RPC is supported in the following environments:

- IBM Software Development Kits (SDKs) Version 6.0
- Sun Java Development Kit (JDK) Version 6.0 that are provided by IBM
- non-IBM SDKs Version 6.0 with this limitation:
  - Xerces limitation on non-IBM SDKs

    If you are using a non-IBM SDK, because of dependencies with the Xerces implementation, you will need to download Xerces-J version 2.6.2 and set it in the classpath before attempting to run the Thin Client for JAX-RPC.
  - Equinox 3.6 OSGi runtime environments

About this task

Note: WS-Addressing is not supported for JAX-RPC web services in an unmanaged client environment. If you need to use WS-Addressing, or a web service standard that relies on WS-Addressing, such as

WS-Notification, you must use the Thin Client for Java API for XML-based Web Services (JAX-WS) instead. To learn how to setup and run the Thin Client for JAX-WS, see the Thin Client for JAX-WS documentation.

Procedure

1. Configure the path. You can add the Java `bin` directories to your path by typing:
2. Configure the classpath.

```
export CLASSPATH=.:<your_web_services_thin_client_install_directory>/com.ibm.ws.webservices.thinclient_8.0.0.jar:
<your_application_jars>;$CLASSPATH
```

   - If you are using a non-IBM SDK, obtain a Xerces `xml-apis.jar` and `xercesImpl.jar` from the Xerces web site and configure the classpath definition.

```
export CLASSPATH=.:<your_Xerces_install_directory>/xml-apis.jar:<your_Xerces_install_directory>
\xercesImpl.jar:$CLASSPATH
```

3. Configure SSL for the client.
   a. Add the following system properties to the Java command:

```
-Dcom.ibm.SSL.ConfigURL=file:///home/sample/ssl.client.props
```

   You can obtain the `ssl.client.props` file from the WebSphere Application Server installation and modify the file to suit your environment. You must, at a minimum, update the location of the `com.ibm.ssl.keyStore` and `com.ibm.ssl.trustStore` key files in the `ssl.client.props` file to the match location of your target environment. For example, use these SSL configuration settings when running the application with a Sun JRE:

```
com.ibm.ssl.protocol=SSL
com.ibm.ssl.trustManager=SunX509
com.ibm.ssl.keyManager=SunX509
com.ibm.ssl.contextProvider=SunJSSE

com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.keyStoreProvider=SUN
com.ibm.ssl.keyStore=/home/user1/etc/key.jks

com.ibm.ssl.trustStoreType=JKS
com.ibm.ssl.trustStoreProvider=SUN
com.ibm.ssl.trustStore=/home/user1/etc/trust.jks
```

   The key store file and trust store file must be created using the Java keytool utility before the application runs. The automatic key file generation is not supported with a non-IBM product JRE.

4. Enter the following command to run your client application:

```
$JAVA_HOME/bin/java -Dcom.ibm.SSL.ConfigURL=file:///home/sample/ssl.client.props <your_client_application>
```

Results

You have set up an unmanaged JAX-RPC client runtime environment that can be used to invoke web services hosted on a WebSphere Application Server.

Running an unmanaged web services JAX-WS client

WebSphere Application Server provides a thin Java Platform, Standard Edition 6 (Java SE 6) web services client runtime implementation that is based on the Java API for XML-based Web Services (JAX-WS) 2.2 specification. The Thin Client for JAX-WS with WebSphere Application Server is a stand-alone Java SE 6 client environment that enables running unmanaged JAX-WS web services client applications in a non-WebSphere environment to invoke web services that are hosted by the application server.

Before you begin

Note: You can use the Thin Client for JAX-WS with WebSphere Application Server as a stand-alone client run time in a pure Java SE environment, or within an OSGi environment. The Thin Client for JAX-WS is not supported running within WebSphere Application Server or WebSphere Application Client environments. In this version of the application server, with the exception of the

Administration Thin Client, other Thin Client run times provided with the application server can also reside in the CLASSPATH and coexist with the Thin Client for JAX-WS.

Before you set up a JAX-WS unmanaged client execution environment, obtain the Thin Client for JAX-WS Java archive (JAR) file. To obtain the Thin Client for JAX-WS, install WebSphere Application Server Version 8.0 or the Application Client for WebSphere Application Server Version 8.0. The Thin Client for JAX-WS JAR file, `com.ibm.jaxws.thinclient_8.0.0.jar`, is located in the *app_server_root*`\runtimes` directory.

Copy the Thin Client for JAX-WS, `com.ibm.jaxws.thinclient_8.0.0.jar` file and the `endorsed_apis_8.0.0.jar` files, to other machines to create a lightweight client environment that enables communications with the product. Copies of the Thin Client for JAX-WS are subject to the same terms and conditions of the license agreement for the WebSphere product where you obtained the Thin Client for JAX-WS. Refer to the license agreements for correct usage and other limitations.

The Thin Client for JAX-WS is supported in the following environments:

- IBM Software Development Kits (SDKs) Version 6.0
- non-IBM SDKs V6.0 with the following limitation:
  - Xerces limitation on non-IBM SDKs

    You must download Xerces-J Version 2.6.2, and add the file to the classpath when setting up the Thin Client for JAX-WS environment.
  - WS-SecurityKerberos on non-IBM SDKs

    WS-SecurityKerberos is not supported with the Sun JDK or other non-IBM SDKs. Applications running in a Thin Client for JAX-WS environment that make use of WS-Security message level protection and use Kerberos security tokens as described in the Web Services Security Kerberos Token Profile 1.1 specification, do not correctly work on non-IBM JDKs. This limitation exists because of a dependancy on the IBM JGSS provider that is only available within IBM SDKs.
- Equinox 3.6 OSGi runtime environments

About this task

Set up a Thin Client for JAX-WS environment by completing the following steps.

Procedure

1. Copy the Thin Client for JAX-WS JAR file, `com.ibm.jaxws.thinclient_8.0.0.jar`, to other machines to create a lightweight client environment.
2. Use the Java Endorsed Standards Override Mechanism to override APIs that are available in the JDK on your system.

   Because the Thin Client for JAX-WS with WebSphere Application ServerVersion 8.0 requires APIs that are more current than what is available in JDKs to support JAX-WS 2.2 and JAXB 2.2 implementations, you must override the default JDK APIs in use by your system by using the Java Endorsed Standards Override Mechanism.

   Copy the *app_server_root*`\runtimes\endorsed\endorsed_apis_8.0.0.jar` file into the default directory, `JAVA_JRE\lib\endorsed`. Alternatively, you can use the `java.endorsed.dirs` property to specify a directory of your choice. If you choose to use an alternative directory, it is a best practice to only include the `endorsed_apis` JAR file.
3. Configure the path. Enter the following command to add the Java `bin` directories to your path:
4. Configure the classpath.
   - Add the Thin Client for JAX-WS JAR file to the classpath definition.

Important: If the Thin Client is to use the Java Message Service (JMS), then **all** the JAR files that are required must be in the classpath for JMS and for the client so that entries exist for all the required files. Otherwise, required files will not be identified as installed and ready for use.

```
export CLASSPATH=.:<your_jax-ws_thin_client_install_directory>/com.ibm.jaxws.thinclient_8.0.0.jar:
<your_application_jars>;$CLASSPATH
```

- If you are using a non-IBM SDK, obtain a Xerces `xml-apis.jar` file and `xercesImpl.jar` file from the Xerces website, and configure the classpath definition.

```
export CLASSPATH=.:<your_Xerces_install_directory>/xml-apis.jar:<your_Xerces_install_directory>
\xercesImpl.jar:$CLASSPATH
```

5. Configure SSL for the client.
   a. Add the following system properties to the Java command:

```
-Dcom.ibm.SSL.ConfigURL=file:///home/sample/ssl.client.props
```

   You can obtain the `ssl.client.props` file from the WebSphere Application Server installation and modify the file to suit your environment. You must, at a minimum, update the location of the `com.ibm.ssl.keyStore` and `com.ibm.ssl.trustStore` key files in the `ssl.client.props` file to the match location of your target environment. For example, use these SSL configuration settings when running the application with a Sun JRE:

```
com.ibm.ssl.protocol=SSL
com.ibm.ssl.trustManager=SunX509
com.ibm.ssl.keyManager=SunX509
com.ibm.ssl.contextProvider=SunJSSE

com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.keyStoreProvider=SUN
com.ibm.ssl.keyStore=/home/user1/etc/key.jks

com.ibm.ssl.trustStoreType=JKS
com.ibm.ssl.trustStoreProvider=SUN
com.ibm.ssl.trustStore=/home/user1/etc/trust.jks
```

   The key store file and trust store file must be created using the Java keytool utility before the application runs. The automatic key file generation is not supported with a non-IBM product JRE.

6. Run your client application:
   - Enter the following command if you have copied the `endorsed_apis_8.0.0.jar` file into the `JAVA_JRE\lib\endorsed` default directory; for example:

```
$JAVA_HOME/bin/java -Dcom.ibm.SSL.ConfigURL=file:///home/sample/ssl.client.props <your_client_application>
```

   - Enter the following command if you have copied the `endorsed_apis_8.0.0.jar` file into a directory other than the default `JAVA_JRE\lib\endorsed` directory; for example:

```
$JAVA_HOME/bin/java
-Djava.endorsed.dirs=<directory_that_includes_endorsed_apis_8.0.0.jar>
-Dcom.ibm.SSL.ConfigURL=file:///home/sample/ssl.client.props <your_client_application>
```

Results

You have set up an unmanaged JAX-WS client runtime environment to invoke web services hosted on a WebSphere Application Server.

Testing web services-enabled clients

Once you have developed, assembled, deployed and configured your web service, you can test to confirm your web service runs in the application server environment.

Before you begin

Before testing your web services Java client to confirm your web service runs in the WebSphere Application Server environment, verify that the server endpoint specified in the client Web Services Description Language (WSDL) file is running and available.

About this task

Tests are run differently depending on whether the client module is in a Java EE container or if the client is running in the Thin Client for Java API for XML-based RPC (JAX-RPC) with WebSphere Application Server application environment or the Thin Client for Java API for XML-Based Web Services (JAX-WS) with WebSphere Application Server application environment.

Procedure

1. Test an unmanaged client JAR file by running your application with the **java** command.

   For JAX-WS applications:

```
"$JAVA_HOME/bin/java"
-Djava.endorsed.dirs=<your_jax-ws_thin_client_install_directory>/endorsed_apis_8.0.0.jar
-classpath
"<your_JAX-WS_thin_client_install_directory>/runtimes/com.ibm.jaxws.thinclient_8.0.0.jar:
<list_of_your_application_jars_and_classes>"
<fully_qualified_class_name_to_run> <your_application_parameters>
```

   The unmanaged client application runs.

2. Test a managed JAX-RPC client EAR file or an unmanaged JAX-WS client EAR file.
   a. Run your client application with the **launchClient** command. The following example illustrates the use of this command:

```
launchClient clientEar
```

Results

You have a web services-enabled client that is tested. Now you can add security measures to the web service. Security measures are optional.

Administering deployed web services applications

You can administer deployed web services applications using the administrative console.

Before you begin

Before you can administer a web service application, you need to deploy your web service application.

About this task

You can use the administrative console to administer Java API for XML-Based Web Services (JAX-WS) service provider or service client applications or Java API for XML-based RPC (JAX-RPC) web services.

Procedure

- Administer service providers. You can administer your service providers using the following ways:
  - View service providers at the cell level using the administrative console. You can view the details of your service provider, manage the policy sets for the service, its endpoints and operations, and assign bindings for the policy set attachment at the cell level.
  - View service providers at the application level using the administrative console. You can view the details of your service provider, manage the policy sets for the service, its endpoints and operations, and assign bindings for the policy set attachment at the application level.
  - Manage policy sets and bindings for service providers. You can view the details of your service provider, manage the policy sets for the service, its endpoints and operations, and assign bindings for the policy set attachment.
  - Manage policy sets and bindings for service providers at the application level using the administrative console. You can manage policy sets for the provider, its endpoints, and operations, and assign bindings for the policy set attachment at the application level.

  - View WSDL document using the administrative console . You can view the WSDL document for your JAX-WS application.
- Administer service clients. You can administer your service clients using the following ways:
  - View service clients at the cell level . Your application server instance can have one or more applications deployed on it that contain service clients. You can view a list of your service clients at the cell level using the administrative console.
  - View service clients at the application level . Your application server instance can have one or more applications deployed on it that contains service clients. You can view the service client names that are referenced in an application.
  - Manage policy sets and bindings for service clients. You can view details of your service client reference, manage the policy sets for the service, its endpoints and operations, and assign bindings for the policy set attachment.
  - Manage policy sets and bindings for service clients at the application level. You can manage policy sets for your service client applications or its service references, endpoints, or operations and assign bindings for the policy set attachment at the application level.
- View the deployment descriptors.. View the web services server and client deployment descriptors for a deployed web services application. You can view the bindings in the deployment descriptors. The deployment descriptors are required for JAX-RPC web services. You can optionally use the `webservices.xml` deployment descriptor to augment or override application metadata specified in annotations within your JAX-WS web services.
- Configure the scope of a web service port..**(JAX-RPC applications only)** When a web service application is deployed into WebSphere Application Server, an instance is created for each application or module. The instance contains deployment information for the web module or enterprise bean module, including implementation scope and client bindings information. There are three levels of scope that you can set: application, session and request.
- Suppress the compensation service Not all web servers are configured to handle SOAP messages containing CoordinationContext elements. WebSphere Application Server allows you to configure a custom property for the compensation service which processes a predefined list of Enterprise Java Beans for which no CoordinationContext should be sent on web service requests.

Overview of service and endpoint listeners

Using the administrative function of this product, you can control throughput by starting and stopping individual service listeners and endpoint listeners. When you stop a service listener, this causes any associated endpoint listeners to stop listening to any new inbound requests and the application server rejects any new incoming requests for that service. Additionally, resources become free that can be used to service requests that are already being processed for the service or to service new incoming requests for other services.

Service listener

The service listener can be started or stopped. When a service listener is stopped, all the endpoints for the service are stopped and new inbound requests are no longer processed. The endpoints for the service send out 404 error message response code to indicate that the service listener is currently stopped and cannot service new inbound requests. When the service listener is started, all the endpoints for the service resume processing of new inbound requests.

Endpoint listener

The endpoint listener can be started or stopped. When a service endpoint listener is stopped, the specific endpoint stops listening to any new inbound request. The requests that are already being processed are not affected. The endpoint sends out a 404 error message response code to indicate that the endpoint is currently stopped and cannot service a new, inbound request. When the endpoint listener is started, the specific endpoint resumes listening on new inbound requests. The state of a particular service endpoint does not impact the listening function of other endpoints for the same service.

Administration of service and endpoint listeners

The administration function of the product is enhanced to support service and endpoint listeners. You can use MBeans such as EndpointManager and the EndpointCentralManager to invoke service and endpoint listeners.

After an application containing a web service is installed, you want to verify that the service is installed correctly. You also want to monitor its service listener state, and update the listener state as needed to control the throughput. One option is to use the collection view of service providers in the administrative console of the product to locate the service provider of interest and observe its listener state.

If you do not want the listener state, then select the service and choose to start or stop the service listener. As the system starts or stops the service listener, the status indicator for the service is updated to show that it is started or stopped. This scenario helps you to throttle back traffic to a specific service as needed but leaves the containing application and other services in the application running. See service providers at the cell level using the administrative console. You can also reference service providers at the application level using the administrative console.

Important: You can only start or stop the service listener using the administrative console.

Another option is to use MBeans. With MBeans, you can invoke the startListener or stopListener operations in the EndpointCentralManager MBean or EndpointManager MBean to start or stop the listener service. The administrative console option does not expose the function of starting or stopping the listening state of a specific endpoint in a service. However, the MBeans option provides this capability. You can use scripting to invoke the MBean operations to start or stop the endpoint listener.

EndpointCentralManager MBean

There is an EndpointCentralManager MBean instance in the deployment manager, AdminAgent, and stand-alone server. The EndpointCentralManager MBean provides the administrative convenience to start and stop the service or endpoint listeners across all the deployment targets such as the cluster members in a cluster. You do not have to know the target servers for the service application.

EndpointManager MBean

There is an EndpointManager MBean instance for each web services application module in a server. This MBean instance is created when the application module is started. The MBean instance is deleted when the module is stopped. The MBean provides the start and stop operations to change the service and endpoint listener state. The MBean can also send a Java Management Extension (JMX) notification whenever the listener state has changed.

Viewing service providers at the cell level using the administrative console

You can use this administrative console page to view and manage your service providers at the cell level.

Before you begin

Before completing this task, you need to install one or more Java API for XML-Based Web Services (JAX-WS).

About this task

You can view a list of your installed service providers, such as JAX-WS web services providers in a cell. You can start or stop the service listener.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation

to learn more about the valid roles for the application server.

Procedure

1. Open the administrative console.
2. In the navigation pane, expand **Services > Service providers**.
3. Locate the web service of interest by name and the associated deployed asset in Deployed Asset column.
4. [Optional] Click the service of interest in the Name column to view the detail of the service and manage the policy sets and bindings for that service.
5. Click **Start Listener** or **Stop Listener** to start or stop the service listener. Starting the service listener makes all the endpoints for the service provider available and ready to receive messages. Stopping the listener makes all the endpoints for the service provider unavailable.

Results

When you finish this task, you have viewed the service providers at the cell level. You have also started or stopped the service listener.

What to do next

Proceed to view service providers at the application level using the administrative console.

Service providers collection at the cell level

Use this page to view and manage service providers at the cell level. Java Application Programming Interface (API) for XML-Based Web Services (JAX-WS) service providers are displayed in this view. Java API for XML Remote Procedure Call (JAX-RPC) service providers are not displayed in this view.

To view a service provider using this administrative console page, perform the following steps:

1. Expand **Services** > **Service Providers**.
2. Locate the service of interest by name in the Name column.
3. Click the service of interest in the Name column.
4. View the detail of the service.

To change the listening state of a service, perform the following steps:

1. Select the service of interest, and click **Start Listener** to start the listener service or click **Stop Listener** to stop the listener service. Starting the service listener makes all the endpoints for the service provider to be available and to receive messages. Stopping the listener makes all the endpoints for the service provider unavailable.
2. [Optional] Select the deployed asset link in the Deployed Asset column to access the deployed asset settings page.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Name:

Specifies the name of the service provider. The service, *QName* (`Java class javax.xml.namespace.QName`), is displayed when you hover your mouse pointer over the Name field.

Type:

Specifies the type of service, such as a JAX-WS web service and Web Services Notification (WSN) client service. You can filter by the type of service. If your service was migrated from the Feature Pack for Web Services, you can only filter by the name of the service.

Deployed Asset:

Represents a Java Platform, Enterprise Edition (Java EE) application or a WS-Notification service point.

Table 243. Type of deployed asset. Indicates the type of deployed asset.

	Java EE application
	WS-Notification Service point application

Status:

Indicates the status for the service listener.

Table 244. Status for the service listener. This table describes the status indicators for the deployed asset, service endpoints and service listener.

	Service endpoints are listening.
	Service endpoints are running, but the listener control is not supported because of the characteristics of this service; for example, the application is installed on a Feature Pack for Web Services server.
	Either the deployed asset is partially started or the service endpoints are listening on some, but not all the target servers.
	The deployed asset is partially started and the listener control is not supported.
	Service endpoints are not listening, but the deployed asset is running.
	Service endpoints are not listening because the deployed asset is not running.
	Service endpoints are not listening because the deployed asset is not running. Listener control is not supported.
	Deployed asset is partially stopped. Service endpoints are not listening.
	Deployed asset is partially stopped. Service endpoints are not listening. Listener control is not supported because of the characteristics of this service; for example, the application is installed on a Feature Pack for Web Services server.
	Status cannot be determined.

Viewing service providers at the application level using the administrative console

You can use this administrative console page to view and manage your service providers at the application level.

Before you begin

Before completing this task, you need to install a Java API for XML-Based Web Services (JAX-WS) web service.

About this task

You can view a list of your service providers, such as JAX-WS web services providers at the application level. You can start the service listener to make all the endpoints for the service to be available and to receive messages. You can also stop the service listener.

Procedure

1. Open the administrative console.

2. To view the service providers, click **Applications > Application Types > WebSphere enterprise applications** >*Service_provider_application_instance* > **Service providers**.
3. [Optional] Click a module that contains a service to view the module information.
4. Depending on the current application status, click **Start Listener** or **Stop Listener** to start or stop the service listener. Starting the service listener makes all the endpoints for the service available and ready to receive messages. Stopping the listener makes all the endpoints for the service unavailable. See Endpoint and service listeners overview.

Results

When you complete this task, you have viewed and managed the service providers at the application level.

What to do next

You can proceed to managing policy sets using the administrative console. You can click on a service provider to manage the policy sets and bindings for that service.

Service providers collection at the application level

Use this page to view and manage service providers at the application level.

To view a service provider using this administrative console page, perform the following steps:

1. Click **Applications > Application Types > WebSphere enterprise applications** >*Service_provider_application_instance* > **Service providers**.
2. Locate the service of interest by name in the Name column.
3. Click the service of interest in the Name column.
4. View the detail of the service.

To change the listening state of a service, perform the following steps:

1. Select the service of interest, and click **Start Listener**, to start the service listener or **Stop Listener**, to stop the service listener. Starting the service listener makes all the endpoints for the service provider to be available and to receive messages. Stopping the listener makes all the endpoints for the service provider to be unavailable.
2. [Optional] Click a module name in the Module column to access the module detail page.

Name:

Specifies the name of the service provider. The service *QName* (`Java class javax.xml.namespace.QName`) is displayed when you hover your mouse pointer over the Name field.

Type:

Specifies the type of service.

Module:

Specifies the name of the module that contains the service.

Status:

Indicates the status for the service listener.

Table 245. Status for the service listener. This table describes the status indicators for the deployed asset, service endpoints and service listener.

	Service endpoints are listening.

Table 245. Status for the service listener (continued). This table describes the status indicators for the deployed asset, service endpoints and service listener.

	Service endpoints are running, but the listener control is not supported because of the characteristics of this service; for example, the application is installed on a Feature Pack for Web Services server.
	Either the deployed asset is partially started or the service endpoints are listening on some, but not all the target servers.
	The deployed asset is partially started and the listener control is not supported.
	Service endpoints are not listening, but the deployed asset is running.
	Service endpoints are not listening because the deployed asset is not running.
	Service endpoints are not listening because the deployed asset is not running. Listener control is not supported.
	Deployed asset is partially stopped. Service endpoints are not listening.
	Deployed asset is partially stopped. Service endpoints are not listening. Listener control is not supported because of the characteristics of this service; for example, the application is installed on a Feature Pack for Web Services server.
	Status cannot be determined.

Viewing the detail of a service provider and managing policy sets using the administrative console

Use this administrative console task to view the detail of your service provider and to manage the policy sets for the service, its endpoints and operations. You can also use this page to manage policy sets and bindings or to access additional information for a Service Component Architecture (SCA) composition unit for service providers.

Before you begin

Before completing this task, you need to install one or more Java API for XML-Based Web Services (JAX-WS) or SCA artifacts.

About this task

You have developed a web service that contains all the necessary artifacts and deployed your web services application into your application server instance. Now, you can attach or detach policy sets and manage the associated bindings.

The policy set information is displayed in the Attached Policy Set column. If a policy set is directly attached, then the policy set name appears; for example, `WS-I RSP` is displayed. If there is no policy set attached, and a policy set is attached at a higher level, then the word *inherited* in parentheses is appended to the policy set name, as the following example demonstrates: `WS-I RSP (inherited)`. If there is no policy set attached directly or at a higher level, then `None` is displayed.

Every attachment of a policy set to a service artifact has an assigned binding. The binding information is displayed in the Binding column. The Binding column can contain the following values:

- Not applicable. There is no policy set attached, either directly or to a higher level service resource.
- *Binding_name* or Default. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. Default is displayed if a policy set is attached directly but the service resource uses the default bindings.
- *Binding_name* (inherited) or Default (inherited). A service resource inherits the bindings from an attachment to a higher level resource.

In Version 7.0 and later, there are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and constrained to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have very limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service providers, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service provider resources that have an attached policy set. See service providers policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings were introduced in Version 7.0. These bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general provider policy set bindings by accessing **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. See defining and managing service client or provider bindings. General provider policy set bindings might also be used for trust service attachments.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. Open the administrative console.
2. In the navigation pane, expand **Services > Service providers >***Service_provider_application_instance* **Service providers**.
3. [Optional] Use the **WSDL document** link under the **Additional Properties** section to view the Web Services Description Language (WSDL) for the service. The Application and Module links provide access to the application and module settings page.

   For SCA service providers, the **Additional Properties** section does not show the WSDL document link, instead, there is a composition unit link. Click **Composition unit** to see the composition unit detail page.
4. Select one or more service, endpoints and operations of interest and view the associated service, endpoints and operations.
5. Do one or more of the following actions:

- Click **Attach**, to attach a policy to a selected service, endpoint or operation.
- Click **Detach**, to detach a policy set from a list of attached policy sets for a service, endpoint or operation.

6. Click **Assign Binding** to select from a list of available bindings for the selected policy set attachment. All the bindings are listed along with the following options:

Table 246. Binding descriptions. Use the descriptions of the default bindings for the selected policy set attachment.

Bindings	Description
Default	Specifies the default binding for the selected service, endpoint or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the Default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the runtime checks to see if the attachment includes a binding. If so, it uses that binding. If not, the runtime checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain
New Application Specific Binding	Select this option to create a new application specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached.
Provider sample	Select this option to use the Provider sample binding.
Provider sample V2	Select this option to use the Provider sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets.
Saml Bearer Provider sample	Select this option to use the Saml Bearer Provider sample. The Saml Bearer Provider sample extends the Provider sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets.
Saml HoK Symmetric Provider sample	Select this option to use the Saml HoK Symmetric Provider sample. The Saml HoK Symmetric Provider sample extends the Provider sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default.

7. To close the drop down list for the assign binding action, click **Assign Binding**.

Results

When you finish this task, a policy set is attached, detached or a binding is assigned to the service artifact.

Example

You have configured a service provider, `EchoService12` in the application instance, WSSampleServicesSei. Now you want to attach the WS-Security policy to the `EchoService12Port` endpoint of the `EchoService12` service provider. First locate **`EchoService12`** in the Services > Service providers collection. Click the **`EchoService12`** service provider. Select the check box for the `columoService12Port` resource. Click **Attach** and select **WSSecurity** default policy from the list. Click **Save**, to save your changes to the master configuration.

What to do next

You can now proceed to manage policy sets and bindings for service providers at the application level using the administrative console.

Service provider settings

Use the Service provider settings page to manage the settings for your service providers. You can attach and detach policy sets to an application, its service, endpoints or operations. You can create new bindings,

or use bindings that you have already created for an attached policy set. You can view or change whether the service provider can share its current policy configuration.

Use the **WSDL document** link to view the Web Services Description Language (WSDL) for the service. The Application and Module links provide access to the application and module settings page.

To view this administrative console page, click **Services** > **Service providers** > ***service_provider_instance***.

You can also view this page by clicking **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_provider_application_instance*** > **Service providers** > ***service_provider_instance***.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Table 247. Button descriptions. Use the buttons to manage policy sets and policy set bindings for a service provider, its endpoints, or operations.

Bindings	Description
Attach Policy Set	Click this button to view a list of policy sets available for attachment to the selected service, endpoint, or operation. Select a policy set from the list to attach and it is attached to the selected service, endpoint, or operation. To close the menu list, click **Attach Policy Set**.
Detach Policy Set	Click this button to detach a policy set from a selected service, endpoint, or operation. After the policy set is detached, if there is no policy set attached to an upper level service resource, the **Attached Policy Set** column displays **None** and the **Binding** column displays **Not Applicable**. If there is a policy set attached to an upper level service resource, the **Attached Policy Set** column displays ***policy_set_name*** **(inherited)** and the binding used for the upper level attachment is applied. The binding name is displayed followed by **(inherited)**.

Table 247. Button descriptions (continued). Use the buttons to manage policy sets and policy set bindings for a service provider, its endpoints, or operations.

Bindings	Description
Assign Binding	Click this button to select from a list of available bindings for the selected policy set attachment. The options include the following: **Default** Specifies the default binding for the selected service reference, endpoint, or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the run time checks to see if the attachment includes a binding. If so, it uses that binding. If not, the run time checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain **New Application Specific Binding** Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached. **Provider sample** Select this option to use the Provider sample binding. **Provider sample V2** Select this option to use the Provider sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets. **Saml Bearer Provider sample** Select this option to use the Saml Bearer Provider sample. The Saml Bearer Provider sample extends the Provider sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets. **Saml HoK Symmetric Provider sample** Select this option to use the Saml HoK Symmetric Provider sample. The Saml HoK Symmetric Provider sample extends the Provider sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default. To close the menu list, click **Assign Binding**.

Service provider:

Specifies the name of the service provider that is displayed.

Policy Set Attachments:

Service/Endpoint/Operation:

Specifies the name of the service provider, endpoint, or operation that is contained in a service.

Attached Policy Set:

Specifies the policy set that is attached to a service provider, endpoint, or operation.

The Attached Policy Set column can contain the following values:

- **None**. No policy set is attached, either directly or to a higher-level service resource.
- ***Policy_set_name***. The name of the policy set that is attached directly to the service resource, for example, `WS-I RSP`.
- ***Policy_set_name* (inherited)**. The name of the policy set that is not attached directly to a service resource, but that is attached to a higher-level service resource.

When the value in the column is a link, click the link to view or change settings about the attached policy set.

Binding:

Specifies the binding configuration that is available for a service provider, endpoint, or operation.

The Binding column can contain the following values:
- **Not applicable**. No policy set is attached, either directly or to a higher-level service resource.
- ***Binding_name*** or **Default**. The binding name is displayed if a policy set is attached directly and an application specific binding or a general binding is assigned, for example, `MyBindings1`. **Default** is displayed if a policy set is attached directly but the service resource uses the default bindings.
- ***Binding_name* (inherited)** or **Default (inherited)**. A service resource inherits the bindings from an attachment to a higher-level resource.

When the value in the Binding column is a link, click the link to view or change settings about the binding.

About policy set bindings

In this release, there are two types of bindings: application-specific bindings and general bindings.

Application-specific bindings

You can create application-specific bindings only at a policy set attachment point. These bindings are specific to, and constrained by, the characteristics of the defined policy. Application-specific bindings can provide configuration for advanced policy requirements such as multiple signatures; however, these bindings are reusable only within an application. Also, application-specific bindings have very limited reuse across policy sets.

When you create an application-specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding, and fully configure the bindings for each policy that you add. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application-specific bindings.

For service providers, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service providers policy sets and bindings collection page, for service provider resources that have an attached policy set. Similarly, for service clients, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service clients policy sets and bindings collection page, for service client resources that have an attached policy set.

General bindings

You can configure general bindings to be used across a range of policy sets and they can be reused across applications and for trust service attachments. Although general bindings are highly reusable, they cannot provide configuration for advanced policy requirements such as multiple signatures. There are two types of general bindings: general provider policy set bindings and general client policy set bindings.

You can create general provider policy set bindings by clicking **Services** > **Policy sets** > **General provider policy set bindings** > **New** in the general provider policy sets panel, or by clicking **Services** > **Policy sets** > **General client policy set bindings** > **New** in the general client policy set and bindings panel. For details about defining and managing service client or provider bindings, see the related links. General provider policy set bindings might also be used for trust service attachments.

Policy Sharing:

Specifies whether the service provider can share its current policy configuration.

The Policy sharing column can contain the following values:

- **Not applicable**. The resource does not have a policy set attached, so there is no policy configuration to share.
- **Disabled**. The policy set of the resource cannot be shared. This is the default setting if a policy set is attached to a service.
- **Enabled**. The policy set of the resource can be shared.

When the value in the column is a link, click the link to view or change settings about how the policy configuration can be shared.

For a service, if the policy set is inherited from the parent application, the policy sharing value is also inherited, and you cannot change it. The value is not a link and it is followed by the term `inherited` in parentheses.

For an endpoint or operation, the value is not a link and it is followed by the term `inherited` in parentheses. The setting is inherited from the parent application or service and you cannot change it.

Managing policy sets and bindings for service providers at the application level using the administrative console

Use this administrative console task to manage policy sets for an application or its services, endpoints, and operations.

Before you begin

Before completing this task, you need to install one or more Java API for XML-Based Web Services (JAX-WS) and attach a policy set to each web service.

About this task

You have developed a web service that contains all the necessary artifacts and deployed your web services application into your application server instance. Now, you can attach or detach policy sets and manage the associated bindings.

The policy set information is displayed in the Attached Policy Set column. If a policy set is directly attached, then the policy set name appears; for example, `WS-I RSP` is displayed. If there is no policy set attached, and a policy set is attached at a higher level, then the word *inherited* in parentheses is appended to the policy set name, as the following example demonstrates: `WS-I RSP (inherited)`. If there is no policy set attached directly or at a higher level, then `None` is displayed.

Every attachment of a policy set to a service artifact has an assigned binding. The binding information is displayed in the Binding column. The Binding column can contain the following values:

- Not applicable. There is no policy set attached, either directly or to a higher level service resource.

- *Binding_name* or Default. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. Default is displayed if a policy set is attached directly but the service resource uses the default bindings.
- *Binding_name* (inherited) or Default (inherited). A service resource inherits the bindings from an attachment to a higher level resource.

In Version 7.0 and later, there are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and constrained to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have very limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service providers, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service provider resources that have an attached policy set. See service providers policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings were introduced in Version 7.0. These bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general provider policy set bindings by accessing **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. See defining and managing service client or provider bindings. General provider policy set bindings might also be used for trust service attachments.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. Open the administrative console.
2. **Applications > Application Types > WebSphere enterprise applications** >*Service_provider_application_instance* > **Service provider policy sets and bindings**.

[Optional] When you are accessing a business-level application, click **Business-level applications (BLA) > *My_BLA* > *BLA_instance*** > **Service provider policy sets and bindings**. Alternatively, click **Services > Service providers > *My_BLA* > *BLA_instance*** > **Service provider policy sets and bindings**.

3. Select the check box next to a service resource of interest.
4. Click **Attach** to attach a policy set to an application, service, endpoint or operation.
5. [Optional] Click **Detach** to detach a policy set from a list of attached policy sets for an application, service provider, endpoint or operation.
6. Click **Assign Binding** to select from a list of available bindings for the selected policy set attachment. All the bindings are listed along with the following options:

Table 248. Binding descriptions. Use the descriptions of the default bindings for the selected policy set attachment.

Bindings	Description
Default	Specifies the default binding for the selected service, endpoint or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the Default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the runtime checks to see if the attachment includes a binding. If so, it uses that binding. If not, the runtime checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain
New Application Specific Binding	Select this option to create a new application specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached.
Provider sample	Select this option to use the Provider sample binding.
Provider sample V2	Select this option to use the Provider sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets.
Saml Bearer Provider sample	Select this option to use the Saml Bearer Provider sample. The Saml Bearer Provider sample extends the Provider sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets.
Saml HoK Symmetric Provider sample	Select this option to use the Saml HoK Symmetric Provider sample. The Saml HoK Symmetric Provider sample extends the Provider sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default.

7. To close the drop down list for the assign binding action, click **Assign Binding**.

Results

When you finish this task, a policy set is attached or detached, and a binding is assigned to the service artifact.

Example

If you have configured a service provider application instance, `app1` and you want, for example, to attach the`Username WSSecurity default` policy set to your application, first locate `app1` application in the **Applications > Application Types > WebSphere enterprise applications**. Click `app1` > **Service provider policy sets and bindings** under the Web Services Properties section. Select the check box next to the `app1` service application. Click **Attach** and select `Username WSSecurity default` policy set. Click **Save**, to save your changes to the master configuration. You are returned to the **Applications > Application Types > WebSphere enterprise applications** page.

To assign a binding to the attached policy set, click app1 > **Service provider policy sets and bindings**. Select the check box next to the app1 service application and click **Assign Binding**. Select **Provider sample** binding from the list. Click **Save**, to save your changes to the master configuration.

What to do next

You can proceed to view service providers at the cell level using the administrative console.

Service provider policy sets and bindings collection

Use this page to attach and detach policy sets to an application, a service provider, its endpoints, or operations. You can select the default bindings, create new application-specific bindings, or use bindings that you created for an attached policy set. You can view or change whether the service provider can share its current policy configuration.

This page provides detail information for an application and its associated web service providers, endpoints, and operations. You can view and manage policy set attachments and bindings information using this page.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***Service_provider_application_instance*** > **Service provider policy sets and bindings**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Table 249. Binding descriptions. Use the descriptions of the default bindings for the selected policy set attachment.

Bindings	Description
Attach Policy Set	Click this button to view a list of policy sets available for attachment to the selected service, endpoint, or operation. Select a policy set from the list to attach and it is attached to the selected service, endpoint, or operation. To close the menu list, click **Attach Policy Set**.
Detach Policy Set	Click this button to detach a policy set from a selected service, endpoint, or operation. After the policy set is detached, if there is no policy set attached to an upper level service resource, the **Attached Policy Set** column displays **None** and the **Binding** column displays **Not Applicable**. If there is a policy set attached to an upper level service resource, the **Attached Policy Set** column displays ***policy_set_name*** **(inherited)** and the binding used for the upper level attachment is applied. The binding name is displayed followed by **(inherited)**.

Table 249. Binding descriptions (continued). Use the descriptions of the default bindings for the selected policy set attachment.

Bindings	Description
Assign Binding	Click this button to select from a list of available bindings for the selected policy set attachment. The options include the following: **Default** Specifies the default binding for the selected service reference, endpoint, or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the run time checks to see if the attachment includes a binding. If so, it uses that binding. If not, the run time checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain **New Application Specific Binding** Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached. **Provider sample** Select this option to use the Provider sample binding. **Provider sample V2** Select this option to use the Provider sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets. **Saml Bearer Provider sample** Select this option to use the Saml Bearer Provider sample. The Saml Bearer Provider sample extends the Provider sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets. **Saml HoK Symmetric Provider sample** Select this option to use the Saml HoK Symmetric Provider sample. The Saml HoK Symmetric Provider sample extends the Provider sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default. To close the menu list, click **Assign Binding**.

Application/Service/Endpoint/Operation:

Specifies the name of the application and the associated service providers, endpoints or operations.

The Application/Service/Endpoint/Operation column lists the service application and the service providers, endpoints, or operations that the application contains.

Attached Policy Set:

Specifies the policy set that is attached to a service provider, endpoint, or operation.

The Attached Policy Set column can contain the following values:

- **None**. No policy set is attached, either directly or to a higher-level service resource.
- ***Policy_set_name***. The name of the policy set that is attached directly to the service resource, for example, WS-I RSP.

- ***Policy_set_name* (inherited)**. The name of the policy set that is not attached directly to a service resource, but that is attached to a higher-level service resource.

When the value in the column is a link, click the link to view or change settings about the attached policy set.

Binding:

Specifies the binding configuration that is available for a service provider, endpoint, or operation.

The Binding column can contain the following values:

- **Not applicable**. No policy set is attached, either directly or to a higher-level service resource.
- ***Binding_name*** or **Default**. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. **Default** is displayed if a policy set is attached directly but the service resource uses the default bindings.
- ***Binding_name* (inherited)** or **Default (inherited)**. A service resource inherits the bindings from an attachment to a higher-level resource.

When the value in the Binding column is a link, click the link to view or change settings about the binding.

About policy set bindings

In this release, there are two types of bindings: application-specific bindings and general bindings.

Application-specific bindings

You can create application-specific bindings only at a policy set attachment point. These bindings are specific to, and constrained by, the characteristics of the defined policy. Application-specific bindings can provide configuration for advanced policy requirements such as multiple signatures; however, these bindings are reusable only within an application. Also, application-specific bindings have very limited reuse across policy sets.

When you create an application-specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding, and fully configure the bindings for each policy that you add. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application-specific bindings.

For service providers, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service providers policy sets and bindings collection page, for service provider resources that have an attached policy set. Similarly, for service clients, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service clients policy sets and bindings collection page, for service client resources that have an attached policy set.

General bindings

You can configure general bindings to be used across a range of policy sets and they can be reused across applications and for trust service attachments. Although general bindings are highly reusable, they cannot provide configuration for advanced policy requirements such as multiple signatures. There are two types of general bindings: general provider policy set bindings and general client policy set bindings.

You can create general provider policy set bindings by clicking **Services** > **Policy sets** > **General provider policy set bindings** > **New** in the general provider policy sets panel, or by clicking **Services** >

Policy sets > **General client policy set bindings** > **New** in the general client policy set and bindings panel. For details about defining and managing service client or provider bindings, see the related links. General provider policy set bindings might also be used for trust service attachments.

Policy Sharing:

Specifies whether the service provider can share its current policy configuration.

The Policy sharing column can contain the following values:

- **Not applicable**. The resource does not have a policy set attached, so there is no policy configuration to share.
- **Disabled**. The policy set of the resource cannot be shared. This is the default setting if a policy set is attached to a service.
- **Enabled**. The policy set of the resource can be shared.

When the value in the column is a link, click the link to view or change settings about how the policy configuration can be shared.

For a service, if the policy set is inherited from the parent application, the policy sharing value is also inherited, and you cannot change it. The value is not a link and it is followed by the term `inherited` in parentheses.

For an endpoint or operation, the value is not a link and it is followed by the term `inherited` in parentheses. The setting is inherited from the parent application or service and you cannot change it.

Viewing WSDL documents for service providers using the administrative console

You can locate and view a Web Services Description Language (WSDL) document from the administrative console.

Before you begin

Before completing this task, you need to install or deploy a Java API for XML Web Services (JAX-WS) application. Read about the JAX-WS application deployment model.

About this task

JAX-WS integration with the WebSphere Application Server provides the option to view a Web Services Description Language (WSDL) document that is associated with your web service using the administrative console.

A web service application that contains all the necessary artifacts has been developed and deployed to the Application Server. Now you can view the WSDL document for each service using the administrative console.

Procedure

1. Open the administrative console.
2. In the navigation pane, expand **Services** > **Service providers**.
3. Select a web service of interest in the **Service providers** collection view.
4. Click the **WSDL document** link to view the WSDL document.

Results

When you finish this task, you have viewed the WSDL document for a service artifact.

What to do next

You can attach and detach policy sets and configure bindings information for your application. See managing policy sets and bindings for service providers at the application level using the administrative console.

Viewing service clients at the cell level using the administrative console

You can view all your service clients at the cell level using the administrative console.

Before you begin

Before completing this task, you need to install one or more Java API for XML-Based Web Services (JAX-WS).

About this task

You can view a list of your service clients, such as JAX-WS Web services clients using the administrative consolel.The Name column displays the service client references. The Deployed Asset column displays the name of the deployed asset that contains the service client.

Procedure

1. Open the administrative console.
2. In the navigation pane, expand **Services** > **Service clients**.
3. View the web service client of interest in the Name column and the associated application in the Deployed Asset column. For JAX-WS (WSN) references, the name of the containing WS-Notification service that is a part of the System Integration (SI) is displayed. Hovering over this name displays the Bus:<busName>.

Results

When you complete this task, you have viewed the service clients at the cell level.

What to do next

Proceed to view service clients at the application level using the administrative console. Click each service to see and manage the policy sets and bindings associated with a particular service.

Service client collection at the cell level

Use this page to view and manage service clients at the cell level. Server-based Java API for XML-Based Web Services (JAX-WS) service clients are the only clients that are displayed in this view. Java API for XML-based RPC (JAX-RPC) service clients are not displayed in this view.

To view this administrative console page, click **Services** > **Service clients**. Select the service name link in the Name column to access the service client settings page. Select the deployed asset link in the Deployed Asset column to access the deployed asset settings page.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Name:

Specifies the name of the service client, which is the name of the service provider that is referenced by the client. The full `QName`, Java class `javax.xml.namespace.QName`, is displayed when you hover the mouse pointer over the **Name** field.

If a web services reference has been defined for a service client, the **Name** field contains the name of the service reference under the corresponding service client. The full name of the service reference in name-value pair format is displayed when you hover the mouse pointer over the **Name** field.

Type:

Specifies the type of service such as a JAX-WS client service and Web Services Notification (WSN) client service. You can filter by the type of service.

Deployed Asset:

Represents a Java Platform, Enterprise Edition (Java EE) application or a WS-Notification service. For JAX-WS or WSN clients, the name of the containing WS-Notification service that is a part of the System Integration (SI) is displayed. Hovering over this name displays the Bus:<busName>.

Table 250. Type of deployed asset and service broker. This table describes the representations for the types of deployed assets.

	Java EE application
	WS-Notification service

Viewing service clients at the application level using the administrative console

You can view your installed service clients for an application using this task.

Before you begin

Before completing this task, you need to install a Java API for XML-based Web Services (JAX-WS) web service.

About this task

You can view a list of your service client references.

Your application server instance can have one or more applications deployed on it that contain service clients. This task enables you to view the service names that are referenced in an application.

Procedure

1. Open the administrative console.
2. View the service clients in an application by expanding **Applications > Application Types > WebSphere enterprise applications** >*Service_client_application_instance* > **Service clients**.

   [Optional] When you are accessing a business-level application, click **Services > Service clients > *BLA_client_instance***.
3. Select a client from the Name column to view the client and to manage the policy sets and bindings associated with it.
4. [Optional] Select a module name from the Module column to access the module detail page.

5. [Optional] In the **Additional Properties** section, click **Composition unit**, to see the composition unit detail page.

Results

When you finish this task, you have viewed service clients at the application level.

What to do next

You can now proceed to managing policy sets using the administrative console.

Service clients collection at the application level

Use this page to view and manage service clients at the application level.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***Service_client_application_instance*** > **Service client**.

Name:

Specifies the name of the service client. The service *QName*, `Java class javax.xml.namespace.QName`, is displayed when you hover your mouse pointer over the **Name** field. If a web services reference has been defined for a service client, the **Name** field contains the name of the service reference under the corresponding service client. The full name of the service reference in name-value pair format is displayed when you hover the mouse pointer over the **Name** field.

Type:

Specifies the type of service such as a Java API for XML-Based Web Services (JAX-WS) web service. You can filter by the type of service.

Module:

Specifies the name of the module that contains the service. Selecting a module name accesses the module detail page.

Viewing detail of a service client and managing policy sets using the administrative console

Use this administrative console task to view the detail of your service client reference and to manage the policy sets for the service, its endpoints and operations.

Before you begin

Before completing this task, you need to install one or more Java API for XML-based Web Services (JAX-WS) web services, and attach a policy set to each web service.

About this task

You have developed a web service that contains all the necessary artifacts and deployed your web services application into your application server instance. Now, you can attach or detach policy sets and manage the associated bindings.

The policy set information is displayed in the Attached Policy Set column. If a policy set is directly attached, then the policy set name appears; for example, `WS-I RSP` is displayed. If there is no policy set attached, and a policy set is attached at a higher level, then the word *inherited* in parentheses is

appended to the policy set name, as the following example demonstrates: `WS-I RSP (inherited)`. If there is no policy set attached directly or at a higher level, then `None` is displayed.

Every attachment of a policy set to a service artifact has an assigned binding. The binding information is displayed in the Binding column. The Binding column can contain the following values:

- Not applicable. There is no policy set attached, either directly or to a higher level service resource.
- *Binding_name* or Default. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. Default is displayed if a policy set is attached directly but the service resource uses the default bindings.
- *Binding_name* (inherited) or Default (inherited). A service resource inherits the bindings from an attachment to a higher level resource.

In Version 7.0 and later, there are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and constrained to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have very limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service clients policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings were introduced in Version 7.0. These bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general client policy set bindings by accessing **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. See defining and managing service client or provider bindings.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. Open the administrative console.
2. In the navigation pane, click **Applications > Application Types > WebSphere enterprise applications** >*Service_client_application_instance* > **Service clients**.
3. Select one or more service, endpoints and operations of interest and view the associated service, endpoints and operations.
4. You can perform any of the following actions:
   - Click **Attach**, to attach a policy set to a selected service, endpoint or operation.
   - Click, **Detach**, to detach a policy set from a list of attached policy sets for a service, endpoint or operation. The service name is the service client reference in the application.
5. Click **Assign Binding** to select from a list of available bindings for the selected policy set attachment. All the bindings are listed along with the following options:

Table 251. Binding descriptions. Use the descriptions of the default bindings for the selected policy set attachment.

Bindings	Description
Default	Specifies the default binding for the selected service client, endpoint, or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the run time checks to see if the attachment includes a binding. If so, it uses that binding. If not, the run time checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain
New Application Specific Binding	Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached.
Client sample	Select this option to use the Client sample binding.
Client sample V2	Select this option to use the Client sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets.
Saml Bearer Client sample	Select this option to use the Saml Bearer Client sample. The Saml Bearer Client sample extends the Client sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets.
Saml HoK Symmetric Client sample	Select this option to use the Saml HoK Symmetric Client sample. The Saml HoK Symmetric Client sample extends the Client sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default.

6. To close the drop down list for the assign binding action, click **Assign Binding**.

Results

When you finish this task, a policy set is attached, detached or a binding is assigned to the service artifact.

Example

You have configured a service client reference, `EchoService12` in the application instance, WSSampleClientSei. Now you want to attach the WSSecurity default policy to the `EchoService12Port` endpoint of the `EchoService12` service client reference. First locate `EchoService12` in the Services > Service clients collection. Click the **`EchoService12`** service client reference. Select the check box for the `EchoService12Port` resource and click **Attach**. Select the **WSSecurity** default policy from the list. Click **Save**, to save your changes to the master configuration.

What to do next

You can now proceed to manage policy sets and bindings for service clients at the application level using the administrative console.

Service client settings

Use this administrative console page to manage the settings for your service clients. You can attach and detach policy sets to a service, its endpoints, or operations. You can select default bindings, create new application-specific bindings, or use existing bindings for an attached policy set. You can view or change whether the client uses the policy of the service provider.

This service client page displays configuration information for a service client and the associated endpoints and operations. You can view and manage policy set attachments, bindings information, and whether the client uses the policy of the service provider.

The Application and Module links provide access to the application and module settings page.

To view this administrative console page, click **Services** > **Service clients** > ***service_client_instance***.

You can also view this page by clicking **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_client_application_instance*** > **Service clients** > ***service_client_instance***.

To attach or detach a policy set or binding, do the following:

1. Select a service client, endpoint, or operation from **Service/Endpoint/Operation**. The **Service/Endpoint/Operation** list is nested, indicating parent-child relationships. When you select a parent, the children automatically inherit the settings of the parent.
2. Click the desired button.

Table 252. Button descriptions. Use the buttons to manage policy sets and policy set bindings for service clients.

Button	Resulting action
Attach Client Policy Set	Click this button to view a list of policy sets available for attachment to the selected service, endpoint, or operation. Select a policy set from the list to attach and it is attached to the selected service, endpoint, or operation. To close the menu list, click **Attach Client Policy Set**.
Detach Client Policy Set	Click this button to detach a policy set from a selected service, endpoint, or operation. After the policy set is detached, if there is no policy set attached to an upper level service resource, the Attached Client Policy Set column displays **None** and the Binding column displays **Not Applicable**. If there is a policy set attached to an upper level service resource, the Attached Client Policy Set column displays ***policy_set_name*** **(inherited)** and the binding used for the upper level attachment is applied. The binding name is displayed followed by **(inherited)**.

Table 252. Button descriptions (continued). Use the buttons to manage policy sets and policy set bindings for service clients.

Button	Resulting action
Assign Binding	Click this button to select from a list of available bindings for the selected policy set attachment. The options include the following: **Default** Specifies the default binding for the selected service, endpoint, or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the run time checks to see if the attachment includes a binding. If so, it uses that binding. If not, the run time checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain **New Application Specific Binding** Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached. **Client sample** Select this option to use the Client sample binding. **Client sample V2** Select this option to use the Client sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets. **Saml Bearer Client sample** Select this option to use the Saml Bearer Client sample. The Saml Bearer Client sample extends the Client sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets. **Saml HoK Symmetric Client sample** Select this option to use the Saml HoK Symmetric Client sample. The Saml HoK Symmetric Client sample extends the Client sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default. To close the menu list, click **Assign Binding**.

Service client:

Specifies the name of the service client that is displayed.

Policy Set Attachments:

Service/Endpoint/Operation:

Specifies the name of the service client, endpoints or operations. The full `QName` (Java class `javax.xml.namespace.QName`) is displayed when you hover the mouse pointer over a service client name.

Attached Client Policy Set:

Specifies the policy set that is attached to the service client, endpoints or operations.

The Attached Client Policy Setcolumn can contain the following values:

- **None**. No policy set is attached, either directly or to a higher level service resource.
- ***policy_set_name***. The name of the policy set that is attached directly to the service resource, for example, `WS-I RSP`.

- ***policy_set_name* (inherited)**. The name of the policy set that is not attached directly to a service resource, but that is attached to a higher level service resource.

When the value in the column is a link, click the link to view or change settings about the attached policy set.

Policies Applied:

Specifies the policies that are applied to the resource.

The Policies Applied column can contain the following values:
- **None**. No policies are applied to the service. This is the default setting if there is no policy set attached to the client.
- **Client only**. The client policy set is applied to the service. This is the default setting if a policy set is attached to the client.
- **Provider only**. The policy configuration of the service provider is applied to the service, as long as the client can support those policies.
- **Client and provider**. A policy that is based on both the client policy set and the policy of the service provider is applied to the service.

When the value in the column is a link, click the link to view or change settings about how the policies are applied.

For a service, if the value in the column is a link followed by the word `inherited` in parentheses, this shows a setting that is inherited from the parent application. You can click the link to change the setting for the service.

For an endpoint or operation, the value is not a link and it is followed by the word `inherited` in parentheses. The setting is inherited from the parent application or service and you cannot change it. If there is no applied policy, the entry in the column is `None`.

Binding:

Specifies the binding information available for a service client, endpoint, or operation.

The Binding column can contain the following values:
- **Not applicable**. There is no policy set attached, either directly or to a higher level service resource.
- ***Binding_name*** or **Default**. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. Default is displayed if a policy set is attached directly but the service resource uses the default bindings.
- ***Binding_name* (inherited)** or **Default (inherited)**. A service resource inherits the bindings from an attachment to a higher level resource.

When the value in the Binding column is a link, click the link to view or change settings about the binding.

About policy set bindings

In this release, there are two types of bindings: application-specific bindings and general bindings.

Application-specific bindings

You can create application-specific bindings only at a policy set attachment point. These bindings are specific to, and constrained by, the characteristics of the defined policy. Application-specific bindings can

provide configuration for advanced policy requirements such as multiple signatures; however, these bindings are reusable only within an application. Also, application-specific bindings have very limited reuse across policy sets.

When you create an application-specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding, and fully configure the bindings for each policy that you add. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application-specific bindings.

For service providers, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service providers policy sets and bindings collection page, for service provider resources that have an attached policy set. Similarly, for service clients, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service clients policy sets and bindings collection page, for service client resources that have an attached policy set.

General bindings

You can configure general bindings to be used across a range of policy sets and they can be reused across applications and for trust service attachments. Although general bindings are highly reusable, they cannot provide configuration for advanced policy requirements such as multiple signatures. There are two types of general bindings: general provider policy set bindings and general client policy set bindings.

You can create general provider policy set bindings by clicking **Services** > **Policy sets** > **General provider policy set bindings** > **New** in the general provider policy sets panel, or by clicking **Services** > **Policy sets** > **General client policy set bindings** > **New** in the general client policy set and bindings panel. Read about defining and managing service client or provider bindings for more information. General provider policy set bindings might also be used for trust service attachments.

Managing policy sets and bindings for services references using the administrative console

Use this administrative console task to manage policy sets and bindings for the service reference, its endpoints, and operations.

Before you begin

Before completing this task, you must install one or more Java API for API for XML-based Web Services (JAX-WS) web services, that contain at least one client service reference.

About this task

You have developed a web service that contains all the necessary artifacts and deployed your web services application into your application server instance. Now, you can attach or detach policy sets and manage the associated bindings.

Note: When you configure the policy set attachments for a service reference, you can override the policy set attachments that are inherited from the service client using the administrative console. You can attach a policy set and binding for a service reference that is different from the policy set attachment for the service client. You can also specify to not attach a policy set to a service reference, even if a policy set is attached to the service client.

The default behavior is that a service reference, and its endpoints and operations, inherits the policy set attachment of the corresponding resources of the service. Service references are only valid for service clients.

Using the administrative console, you can configure the service reference to either inherit policy set and bindings configuration from the service client or to specify individual settings for the service reference by attaching policy sets and bindings that are different from the policy sets and bindings attached to the service client.

The policy set information is displayed in the Attached Policy Set column. If a policy set is directly attached, then the policy set name is displayed; for example, `WS-I RSP`. If there is no policy set attached, and a policy set is attached at a higher level or to the service client, then the word *inherited* in parentheses is appended to the policy set name, as the following example demonstrates: `WS-I RSP (inherited)`. If there is no policy set attached directly or at a higher level, then `None` is displayed.

Every attachment of a policy set to a service artifact has an assigned binding. The binding information is displayed in the Binding column. The Binding column can contain the following values:

- Not applicable. There is no policy set attached, either directly, to a service client resource, or to a higher level service reference resource.
- *Binding_name* or Default. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. Default is displayed if a policy set is attached directly but the service reference resource uses the default bindings.
- *Binding_name* (inherited) or Default (inherited). A service resource inherits the bindings from an attachment to a service client resource or a higher level service reference resource.

There are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and constrained to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have very limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high-level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service clients policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings

- General client policy set bindings

You can create general client policy set bindings by accessing **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. See defining and managing service client or provider bindings.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Note: In a mixed cell environment, the following limitations apply to service reference attachments or resource attachments that are specified in name-value pair format:

- You must not create these types of attachments for applications that are deployed on an application server that is prior to WebSphere Application Server Version 8. Service reference attachments are only supported on WebSphere Application Server V8 and later.
- An application that contains these types of attachments must not be deployed on an application server that is prior to WebSphere Application Server Version 8.
- If an application that is deployed in a cluster environment contains these types of attachments, you must not add a member application server that is prior to WebSphere Application Server Version 8 to the cluster.

Procedure

1. Open the administrative console.
2. In the navigation pane, click **Applications > Application Types > WebSphere enterprise applications** >*Service_client_application_instance* > **Service clients**.
3. Select a service references and view the associated service reference, endpoints, and operations.
4. You can perform any of the following actions:
   - Click **Inherit**, to clear existing policy set and binding settings for the service reference and to use policy set attachments that are defined by the service client. By default, a service reference, and its endpoints and operations, inherits the policy set attachment of the corresponding resources of the service.
   - Click **Override**, to override existing policy set and binding settings for the service client and to either define separate policy sets and bindings for the service reference or to specify that a policy set is not attached to the service reference.
   - Click **Attach Client Policy Set**, to attach a policy set to a selected service reference, endpoint, or operation. This button is active only after you have clicked the **Override** button.
   - Click, **Detach Client Policy Set**, to detach a policy set from a list of attached policy sets for a service reference, endpoint, or operation. The service name is the service client reference in the application. This button is active only after you have clicked the **Override** button.
5. Click **Assign Binding** to select from a list of available bindings for the selected policy set attachment. This button is active only after you have clicked the **Override** button. All the bindings are listed along with the following options:

Table 253. Binding descriptions. Use the descriptions of the default bindings to determine which binding to apply to service references.

Bindings	Description
Default	Specifies the default binding for the selected service reference, endpoint, or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the run time checks to see if the attachment includes a binding. If so, it uses that binding. If not, the run time checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain
New Application Specific Binding	Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached.
Client sample	Select this option to use the Client sample binding.
Client sample V2	Select this option to use the Client sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets.
Saml Bearer Client sample	Select this option to use the Saml Bearer Client sample. The Saml Bearer Client sample extends the Client sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets.
Saml HoK Symmetric Client sample	Select this option to use the Saml HoK Symmetric Client sample. The Saml HoK Symmetric Client sample extends the Client sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default.

6. To close the drop-down list for the assign binding action, click **Assign Binding**.
7. (optional) Display inherit policy set attachments confirmation. Selecting this check box enables the inherit policy set attachments confirmation panel. You must expand the Preferences section to select this check box.

Results

When you finish this task, you have specified policy sets and bindings for a service reference.

Example

Suppose that you have configured a service client reference, `EchoService12` in the application instance, WSSampleClientSei. You want to attach the WSSecurity default policy to the `EchoService12Port` endpoint of the `EchoService12` service client reference. You must override the current service client policy set and bindings. Complete the following steps:

1. Locate `EchoService12` in the **Applications > Application Types > WebSphere enterprise applications** >*WSSampleClientSei* > **Service clients** collection. Alternatively, you can locate `EchoService12` in the **Services > Service clients** collection.
2. Click **Override**, to override the service client attachments.
3. Select the check box for the `EchoService12Port` resource, and click **Attach Client Policy Set**.
4. Select the **WSSecurity default** policy from the list.
5. Click **Save** to save your changes to the master configuration.

Suppose later, you want your service reference, `EchoService12` to not use the WSSecurity default policy. Instead, you want to use the policy sets and attachments from the service client, `WSSampleClientSei`. Complete the following steps:

1. Locate `EchoService12` in the **Applications > Application Types > WebSphere enterprise applications** >*WSSampleClientSei* > **Service clients** collection. Alternatively, you can locate `EchoService12` in the **Services > Service clients** collection.
2. Click **Inherit**, to clear the existing policy set and binding settings for the service reference and to use policy set attachments that are defined by the service client.
3. Click **OK** on the Inherit policy sets page to confirm that you want to inherit the policy set attachments that are defined by the service client. You can optionally select the check box to not show the inherit confirmation page in the future.
4. Click **Save** to save your changes to the master configuration.

What to do next

You can now proceed to manage other service references for the service client or to manage policy sets and bindings for service clients at the application level using the administrative console.

Service reference settings

Use this administrative console page to manage the settings for your service references. Service references can inherit the policy sets and bindings of the service client or you can specify policy sets and bindings that are different from those of the service client.

This service references page displays configuration information for a service reference and the associated endpoints and operations. You can view and configure the service reference to either inherit policy set and binding configurations from the service client or to specify individual settings for the service reference by attaching policy sets and bindings that are different from the policy sets and bindings that are attached to the service client.

The Application, Module, and Service client links provide access to the application, module, and service client settings page.

To view this administrative console page, click **Services** > **Service clients** > ***service_reference_instance***.

You can also view this page by clicking **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_client_application_instance*** > **Service clients** > ***service_reference_instance***.

You can specify a policy set and binding for a service reference that is different from the policy set attachment for the service. In addition, you can indicate to not attach a policy set to a service reference, even if a policy set is attached to the service. The default behavior is that a service reference, and its endpoints and operations, inherits the policy set attachment of the corresponding resources of the service. Service references are only valid for the client attachment type.

To attach or detach a policy set or binding, complete the following actions:

1. Select a service reference, endpoint, or operation from **Service Reference/Endpoint/Operation**. The **Service Reference/Endpoint/Operation** list is nested, indicating parent-child relationships. When you select a parent, the children automatically inherit the settings of the parent.
2. Click the button for the action that you want to complete.

Table 254. Button descriptions. Use the buttons to manage policy sets and policy set bindings for service references.

Button	Resulting action
Inherit	Click this button to clear the existing policy set and binding settings for the service reference and to use policy set attachments that are defined by the service client. By default, a service reference, as well as its endpoints and operations, inherits the policy set attachment of the corresponding resources of the service.
Override	Click this button to override the existing policy set and binding settings for the service client and to either define separate policy sets and bindings for the service reference or to specify that a policy set is not attached to the service reference.
Attach Client Policy Set	Click this button to view a list of policy sets available for attachment to the selected service reference, endpoint, or operation. Select a policy set from the list to attach; that policy set is attached to the selected service, endpoint, or operation. To close the menu list, click **Attach Client Policy Set**. This button is active only after you have clicked the **Override** button.
Detach Client Policy Set	Click this button to detach a policy set from a selected service reference, endpoint, or operation. After the policy set is detached, if there is no policy set attached to an upper-level service resource, the Attached Client Policy Set column displays **None** and the Binding column displays **Not Applicable**. If there is a policy set attached to an upper-level service reference resource or service client, the Attached Client Policy Set column displays ***policy_set_name*** **(inherited)** and the binding used for the upper-level attachment is applied. The binding name is displayed followed by **(inherited)**. This button is active only after you have clicked the **Override** button.
Assign Binding	Click this button to select from a list of available bindings for the selected policy set attachment. The options include the following: **Default** Specifies the default binding for the selected service reference, endpoint, or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level for a particular server or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this **Assign Binding** action, the default specified at the nearest scope is used. For any policy set attachment, the runtime environment verifies whether the attachment includes a binding. If so, it uses that binding. If not, the runtime environment checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain **New Application Specific Binding** Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached. **Client sample** Select this option to use the Client sample binding. **Client sample V2** Select this option to use the Client sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets. **Saml Bearer Client sample** Select this option to use the Saml Bearer Client sample. The Saml Bearer Client sample extends the Client sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets. **Saml HoK Symmetric Client sample** Select this option to use the Saml HoK Symmetric Client sample. The Saml HoK Symmetric Client sample extends the Client sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default. To close the menu list, click **Assign Binding**. This button is active only after you have clicked the **Override** button.

Display inherit policy set attachments confirmation: Selecting this check box specifies that you want to enable the inherit policy set attachments confirmation. You must expand the Preferences section to select this check box.

Service Reference/Endpoint/Operation:

Specifies the name of the service reference, endpoints, or operations. The full resource name is displayed when you hover the mouse pointer over a service reference, endpoint or operation.

Client Policy Set:

Specifies the policy set that is attached to the service reference, endpoints, or operations.

The Attached Client Policy Set column can contain the following values:

- **None**. No policy set is attached, either directly to the service client, or to a higher-level service reference resource.
- ***policy_set_name***. The name of the policy set that is attached directly to the service reference resource, for example, `WS-I RSP`.
- ***policy_set_name*** **(inherited)**. The name of the policy set that is not attached directly to a service reference resource; the policy set is attached to a higher-level service reference resource or to a service client resource.

When the value in the column is a link, click the link to view or change settings about the attached policy set.

Policies Applied:

Specifies the policies that are applied to the service reference.

The Policies Applied service reference link is only enabled after you have clicked the Override button.

The Policies Applied column can contain the following values:

- **None**. No policies are applied to the service. This is the default setting if there is no policy set attached to the client.
- **Client only**. The client policy set is applied to the service. This is the default setting if a policy set is attached to the client.
- **Provider only**. The policy configuration of the service provider is applied to the service, as long as the client can support those policies.
- **Client and provider**. A policy that is based on both the client policy set and the policy of the service provider is applied to the service.

When the value in the column is a link, click the link to view or change settings about how the policies are applied.

For an endpoint or operation, the value is not a link and it is followed by the word `inherited` in parentheses. The setting is inherited from the parent application or service and you cannot change it. If there is no applied policy, the entries in the column are **None** or **None (inherited)**.

Binding:

Specifies the binding information available for a service reference, endpoint, or operation.

The Binding column can contain the following values:

- **Not applicable**. There is no policy set attached, either directly to the service client, or to a higher-level service reference resource.
- ***Binding_name*** or **Default**. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. Default is displayed if a policy set is attached directly but the service reference resource uses the default bindings.
- ***Binding_name*** **(inherited)** or **Default (inherited)**. A service resource inherits the bindings from an attachment to a higher-level service reference resource or to a service client resource.

About policy set bindings

In this release, there are two types of bindings: application-specific bindings and general bindings.

Application-specific bindings

You can create application-specific bindings only at a policy set attachment point. These bindings are specific to, and constrained by, the characteristics of the defined policy. Application-specific bindings can provide configuration for advanced policy requirements such as multiple signatures; however, these bindings are reusable only within an application. Also, application-specific bindings have very limited reuse across policy sets.

When you create an application-specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding, and fully configure the bindings for each policy that you add. For WS-Security policy, some high-level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application-specific bindings.

For service providers, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service providers policy sets and bindings collection page, for service provider resources that have an attached policy set. Similarly, for service clients, you can create application-specific bindings only by selecting **Assign Binding** > **New Application Specific Binding**, on the Service clients policy sets and bindings collection page, for service client resources that have an attached policy set.

General bindings

You can configure general bindings to be used across a range of policy sets and they can be reused across applications and for trust service attachments. Although general bindings are highly reusable, they cannot provide configuration for advanced policy requirements such as multiple signatures. There are two types of general bindings: general provider policy set bindings and general client policy set bindings.

You can create general provider policy set bindings by clicking **Services** > **Policy sets** > **General provider policy set bindings** > **New** in the general provider policy sets panel, or by clicking **Services** > **Policy sets** > **General client policy set bindings** > **New** in the general client policy set and bindings panel. Read about defining and managing service client or provider bindings for more information. General provider policy set bindings might also be used for trust service attachments.

Managing policy sets and bindings for service clients at the application level using the administrative console

Use this administrative console task to manage policy sets for service clients applications or its services, endpoints, or operations.

Before you begin

Before completing this task, you need to install one or more Java API for XML-Based Web Services (JAX-WS) applications.

About this task

You have developed a web service that contains all the necessary artifacts and deployed your web services application into your application server instance. Now, you can attach or detach policy sets and manage the associated bindings.

The policy set information is displayed in the Attached Policy Set column. If a policy set is directly attached, then the policy set name appears; for example, `WS-I RSP` is displayed. If there is no policy set attached, and a policy set is attached at a higher level, then the word *inherited* in parentheses is appended to the policy set name, as the following example demonstrates: `WS-I RSP (inherited)`. If there is no policy set attached directly or at a higher level, then `None` is displayed.

Every attachment of a policy set to a service artifact has an assigned binding. The binding information is displayed in the Binding column. The Binding column can contain the following values:

- Not applicable. There is no policy set attached, either directly or to a higher level service resource.
- *Binding_name* or Default. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. Default is displayed if a policy set is attached directly but the service resource uses the default bindings.
- *Binding_name* (inherited) or Default (inherited). A service resource inherits the bindings from an attachment to a higher level resource.

In Version 7.0 and later of this product, there are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and constrained to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have very limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service clients, you can only create application specific bindings by selecting **Assign Binding** > **New Application Specific Binding** for service client resources that have an attached policy set. See service clients policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding** > **New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings were introduced in Version 7.0 of this product. These bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service

attachments. Though general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general client policy set bindings by accessing **Services** > **Policy sets** > **General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services** > **Policy sets** > **General client policy set bindings** > **New** in the general client policy set and bindings panel. See defining and managing service client or provider bindings.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. Open the administrative console.
2. In the navigation pane, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***Service_client_application_instance*** > **Service client policy sets and bindings**.
3. Select the check box next to the **Application/Service/Endpoint/Operation** column. The service name is the service client reference in the application.
4. Click **Attach** to attach a policy set to an application, service, endpoint or operation.
5. [Optional] Click **Detach** to detach a policy set from a list of attached policy sets for an application, service client, endpoint or operation.
6. Click **Assign Binding** to select from a list of available bindings for the selected policy set attachment. All the bindings are listed along with the following options:

Table 255. Binding descriptions. Use the descriptions of the default bindings for the selected policy set attachment.

Bindings	Description
Default	Specifies the default binding for the selected service client, endpoint, or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the run time checks to see if the attachment includes a binding. If so, it uses that binding. If not, the run time checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain in which the server resides 3. Default general bindings for the global security domain
New Application Specific Binding	Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached.
Client sample	Select this option to use the Client sample binding.
Client sample V2	Select this option to use the Client sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets.
Saml Bearer Client sample	Select this option to use the Saml Bearer Client sample. The Saml Bearer Client sample extends the Client sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets.
Saml HoK Symmetric Client sample	Select this option to use the Saml HoK Symmetric Client sample. The Saml HoK Symmetric Client sample extends the Client sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default.

Results

When you finish this task, a policy set is attached or detached, and a binding is assigned to the service artifact.

Example

If you have configured a service client application instance, `app1` and you want, for example, to attach the Username WSSecurity default policy set to your application, first locate `app1` application in the **Applications** > **Application Types** > **WebSphere enterprise applications**. Click ***app1*** > **Service client policy sets and bindings** under the Web Services Properties section. Select the check box next to the `app1` service application. Click **Attach** and select **Username WSSecurity default** policy set. Click **Save**, to save your changes to the master configuration.

To assign a binding to the attached policy set, click ***app1*** > **Service client policy sets and bindings**. Select the check box next to the `app1` service application and click **Assign Binding**. Select **client sample** binding from the list. Click **Save**, to save your changes to the master configuration.

What to do next

You can proceed to view service clients at the cell level using the administrative console.

Service client policy set and bindings collection

Use this page to attach and detach policy sets to an application, a service client, its endpoints, or operations. You can select the default bindings, create new application-specific bindings, or use existing bindings for an attached policy set. You can view or change whether the client uses the policy of the service provider.

This page displays detail information for an application and its associated web service clients, endpoints, and operations. You can view and manage policy set attachments and bindings information using this page.

To view this administrative console page, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_client_application_instance*** > **Service client policy sets and bindings**.

This console page can also be viewed for WS-Notification service clients by clicking one of the following paths:

- **Service integration** > **WS-Notification** > **Services** > ***service_name*** > **[Additional properties] Outbound request policy sets and bindings**
- **Service integration** > **Buses** > ***bus_name*** > **[Services] WS-Notification services** > ***service_name*** > **[Additional properties] Outbound request policy sets and bindings**

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Table 256. Button descriptions. Use the buttons to manage policy sets and policy set bindings for a service client, its endpoints, or operations.

Button	Resulting action
Attach Client Policy Set	Click this button to view a list of policy sets available for attachment to the selected service, endpoint, or operation. Select a policy set from the list to attach and it is attached to the selected service, endpoint, or operation. To close the menu list, click **Attach Client Policy Set**. **Note:** Attach policy sets at the highest level, the EAR server level for example, and let the lower levels inherit those bindings. This can significantly improve the processing time needed to attach sets to multiple operations. After you attach sets at the highest level, you can then customize the lower levels by detaching sets or removing bindings from those specific operations.
Detach Client Policy Set	Click this button to detach a policy set from a selected service, endpoint, or operation. After the policy set is detached, if there is no policy set attached to an upper level service resource, the Attached Client Policy Set column displays **None** and the Binding column displays **Not Applicable**. If there is a policy set attached to an upper level service resource, the Attached Client Policy Set column displays ***policy_set_name*** **(inherited)** and the binding used for the upper level attachment is applied. The binding name is displayed followed by **(inherited)**.
Assign Binding	Click this button to select from a list of available bindings for the selected policy set attachment. All the bindings are listed along with the following options: **Default** Specifies the default binding for the selected service, endpoint or operation. You can specify client and provider default bindings to be used at the cell level or global security domain level, for a particular server, or for a security domain. The default bindings are used when an application-specific binding has not been assigned to the attachment. When you attach a policy set to a service resource, the binding is initially set to the default. If you do not specifically assign a binding to the attachment point using this Assign Binding action, the default specified at the nearest scope is used. For any policy set attachment, the run time checks to see if the attachment includes a binding. If so, it uses that binding. If not, the run time checks in the following order and uses the first available default binding: 1. Default general bindings for the server 2. Default general bindings for the domain that the server resides 3. Default general bindings for the global security domain **New Application Specific Binding** Select this option to create a new application-specific binding for the policy set attachments. The new binding you create is used for the selected resources. If you select more than one resource, ensure that all selected resources have the same policy set attached. **Client sample** Select this option to use the Client sample binding. **Client sample V2** Select this option to use the Client sample V2 binding when you are using either the Kerberos V5 WSSecurity default or the TrustServiceKerberosDefault policy sets. **Saml Bearer Client sample** Select this option to use the Saml Bearer Client sample. The Saml Bearer Client sample extends the Client sample binding to support SAML Bearer token usage scenarios. You can use this sample with any of the SAML bearer token default policy sets. **Saml HoK Symmetric Client sample** Select this option to use the Saml HoK Symmetric Client sample. The Saml HoK Symmetric Client sample extends the Client sample binding to support SAML holder-of-key (HoK) symmetric key token usage scenarios. You can use this sample with one of the SAML HoK Symmetric key default policy sets: either SAML11 HoK Symmetric WSSecurity default or SAML20 HoK Symmetric WSSecurity default. To close the menu list, click **Assign Binding**. **Note:** Assign bindings at the highest level, the EAR server level for example, and let the lower levels inherit those bindings. This can significantly improve the processing time needed to attach sets and bindings to multiple operations. After you assign bindings at the highest level, you can then customize the lower levels by detaching sets or removing bindings from those specific operations.

Application/Service/Endpoint/Operation:

Specifies the name of the application and the associated service client, endpoints, or operations. For WS-Notification service clients, the first entry is associated with the WS-Notification service, not an application.

Attached Client Policy Set:

Specifies the policy set that is attached to the application, service clients, endpoints, or operations.

The Attached Client Policy Set column can contain the following values:

- None. No policy set is attached directly, or is attached at an upper level.
- ***policy_set_name***. The name of the policy set that is directly attached, for example, WS-I RSP.
- *policy_set_name* (inherited). A policy set is not directly attached to the resource, but a policy set is attached to a higher-level resource.

When the value in the column is a link, click the link to view or change settings about the attached policy set.

Policies Applied:

Specifies the policies that are applied to the resource. This column is not applicable and is not shown for WS-Notification service clients.

The Policies Applied column can contain the following values:

- **None**. No policies are applied to the application or service. This is the default setting if there is no policy set attached to the client.
- **Client only**. The client policy set is applied to the application or service. This is the default setting if a policy set is attached to the client.
- **Provider only**. The policy configuration of the service provider is applied to the application or service, as long as the client can support those policies.
- **Client and provider**. A policy that is based on both the client policy set and the policy of the service provider is applied to the application or service.

When the value in the column is a link, click the link to view or change settings about how the policies are applied.

For a service, if the value in the column is a link followed by the word `inherited` in parentheses, this shows a setting that is inherited from the parent application. You can click the link to change the setting for the service.

For an endpoint or operation, the value is not a link and it is followed by the word `inherited` in parentheses. The setting is inherited from the parent application or service and you cannot change it.

Binding:

Specifies the name of the binding associated with a policy set.

The Binding column can contain the following values:

- **Not applicable**. There is no policy set attached, either directly or to a higher-level service resource.
- ***Binding_name*** or **Default**. The binding name is displayed if a policy set is attached directly and an application-specific binding or a general binding is assigned, for example, `MyBindings1`. **Default** is displayed if a policy set is attached directly but the service resource uses the default bindings.
- ***Binding_name*** **(inherited)** or **Default (inherited)**. A service resource inherits the bindings from an attachment to a higher-level resource.

When the value in the Binding column is a link, click the link to view or change settings about the binding.

About policy set bindings

In this release, there are two types of bindings: application-specific bindings and general bindings.

Application-specific bindings

You can create application-specific bindings only at a policy set attachment point. These bindings are specific to, and constrained by, the characteristics of the defined policy. Application-specific bindings can provide configuration for advanced policy requirements such as multiple signatures; however, these bindings are reusable only within an application. Also, application-specific bindings have very limited reuse across policy sets.

When you create an application-specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding, and fully configure the bindings for each policy that you add. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application-specific bindings.

For service providers, you can create application-specific bindings only by selecting **Assign Binding > New Application Specific Binding**, on the Service providers policy sets and bindings collection page, for service provider resources that have an attached policy set. Similarly, for service clients, you can create application-specific bindings only by selecting **Assign Binding > New Application Specific Binding**, on the Service clients policy sets and bindings collection page, for service client resources that have an attached policy set.

General bindings

You can configure general bindings to be used across a range of policy sets and they can be reused across applications and for trust service attachments. Although general bindings are highly reusable, they cannot provide configuration for advanced policy requirements such as multiple signatures. There are two types of general bindings: general provider policy set bindings and general client policy set bindings.

You can create general provider policy set bindings by clicking **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel, or by clicking **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. For details about defining and managing service client or provider bindings, see the related links. General provider policy set bindings might also be used for trust service attachments.

Viewing web services deployment descriptors in the administrative console

You can view the web services client and server deployment descriptors for a deployed web services application. You can view the bindings in the deployment descriptors.

Before you begin

Before you can view the deployment descriptors, you need to complete all tasks required to implement a JAX-RPC web service, create the deployment descriptor templates that were generated by the WSDL2Java command-line tool, configure the deployment descriptors and bindings, and deploy the Web service application into WebSphere Application Server. Review the implementing web services application documentation for more information on developing and implementing web services.

About this task

Deployment descriptors contain the information that is needed by a web services client to communicate with the server for which the Web services is installed. This information is added to the deployment descriptor templates after a web service is developed or an existing web service is located. The data that you can view in the deployment descriptor includes the following:

- The web service description including the name, WSDL file, WSDL file location and the mapping file.
- The port description, including the port component name, the WSDL port, the service endpoint interface that indicate the service's bindings, and the EJB that is used to implement the web service.

After you have developed a web service that contains all the necessary artifacts, created the deployment descriptors from the deployment descriptor templates, configured the deployment descriptors, and deployed the web services application into WebSphere Application Server; now you can view the deployment descriptors and bindings in the administrative console.

Similar to Java API for XML-based RPC (JAX-RPC) web services, you can use deployment descriptors to describe JAX-WS web services. For JAX-WS web services, the use of the `webservices.xml` deployment descriptor is optional because you can use annotations to specify all of the information that is contained within the deployment descriptor file. You can use the deployment descriptor file to augment or override existing JAX-WS annotations. Any information that you define in the `webservices.xml` deployment descriptor overrides any corresponding information that is specified by annotations.

Procedure

1. Open the administrative console.
2. Click **Applications > Application Types > WebSphere enterprise applications > *application_name* > Manage Modules**.
   - Click **View web services client deployment descriptor extension**.
   - Click **View web services server deployment descriptor**.
   - Click **View web services server deployment descriptor extension**.
3. Click **Expand All** to view the deployment descriptor contents.
4. Verify deployment descriptor and bindings configurations.

What to do next

You have viewed and verified the deployment descriptors and bindings for the web services application.

Configuring the scope of a JAX-RPC web services port

When a Java API for XML-based RPC (JAX-RPC) web service application is deployed into WebSphere Application Server, an instance is created for each application or module. The instance contains deployment information for the web module or enterprise bean module, including implementation scope, client bindings and deployment descriptor information. There are three levels of scope that can be set: application, session and request.

Before you begin

Deploy a web service into the WebSphere Application Server. To learn more, read about deploying web services applications onto application servers.

About this task

The Web Services for Java Platform, Enterprise Edition (Java EE) specification states that web services implementations must be stateless. Therefore, to maintain specification compliance, the scope can remain at the application level because the state relevant to the individual sessions level or the requests level is

not supposed to be maintained in the implementation. If you want to deviate from the specification and want to access a different JavaBeans instance, because you are looking for information that is located in another JavaBeans implementation, the scope settings need to change.

The setting that you configure for the scope determines how frequently a new instance of a service implementation class is created for the web services ports in a module. Use this task to configure the scope of a web services port.

This task applies only to Java API for XML-based RPC (JAX-RPC) web services.

To change the scope setting in the administrative console:

Procedure

1. Open the administrative console.
2. Click **Applications > Application Types > WebSphere enterprise applications *application_name* > Manage Modules > *module instance* > Web services implementation scope**
3. Set the scope to application, session or request. The application scope causes the same instance of the implementation to be used for all requests on the application. The session scope causes the same instance to be used for all requests in each session. The request scope causes a new instance to be used for every request. For example, with the scope set to application, every message that comes to the server accesses the same JavaBeans instance because that is the way the scope settings are configured.
4. Click **Apply**.
5. Click **OK**.

Results

The scope for a web services port is configured.

What to do next

Now you can finish any other configurations, start or stop the application, and verify the expected behavior of your web service.

Web services implementation scope

Use this page to view and manage the scope of the ports of a Web service application.

To view this administrative console page, click **Applications > Application Types > WebSphere enterprise applications > *application_name* > Manage Modules > *module_instance* > Web services implementation scope**.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

Port:

Specifies a port name for a web service. A module can contain one or more web services, each of which can contain one or more ports.

Web service:

Specifies the name of the web service. A module can contain one or more web services.

URI:

Specifies the Uniform Resource Identifier (URI) of the binding file that defines the scope. The URI is relative to the web module.

Scope:

Specifies the scope of a port. The valid values for scope are request, session and application.

The scope value determines when a new instance of a service implementation is created for the web service ports in a module. When the scope value is set to application, the same instance of the implementation is used for all requests on the application. When the scope value is set to session, the same instance is used for all requests on each session. When the scope value is set to request, a new instance is created for every request.

Suppressing the compensation service

Not all web servers are configured to handle SOAP messages containing CoordinationContext elements. You can use WebSphere Application Server to configure a custom property for the compensation service which processes a predefined list of Enterprise Java Beans for which no CoordinationContext should be sent on web service requests.

About this task

When the compensation service is used, CoordinationContext elements are included in the outgoing SOAP header. For example:

```
<wscoor:CoordinationContext soapenv:mustUnderstand="1"
...
</wscoor:CoordinationContext>
```

If such a SOAP message is received by a web server which is not configured to process CoordinationContext elements, an exception message is produced. See the following example:

```
Header block local name 'CoordinationContext' is not defined.
```

You can construct a file containing a list of all Enterprise Java Beans which should not send the CoordinationContext element in web service requests. This file must be in plain text format and must contain one entry per line, in the following format:

```
application_name#module#bean
application_name#module#bean
application_name#module#bean
```

Here `application_name` is the name of the application as known on the server; `module` is the name of the Enterprise Java Bean jar; and `bean` is the name of the Enterprise Java Bean.

Note: This file must only contain entries for beans not configured to use the compensation service. This custom property will not be effective for any beans listed in the file which have compensation service metadata associated with them.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Compensation Service** > **[Additional Properties] Custom Properties**
3. Click **New**.
4. Enter SUPPRESS_CSCOPE_ON_WS_CALLS in the Name field.
5. In the Value field, enter a fully qualified file name.
6. Click **Apply** or **OK**.
7. Click **Save** to save your changes to the master configuration.

8. Restart the server.

Results

Web service requests sent from Enterprise Java Beans listed in the custom property file will not contain CoordinationContext metadata in the outgoing SOAP message header.

Managing policy sets using the administrative console

You can use policy sets, or assertions that define services, to simplify your web services configuration because policy sets group security and other web services settings into reusable units. You can use the administrative console to create, modify, and delete custom policy sets.

Before you begin

Before creating policy sets, first identify the security and other requirements of the web service.

Note: You can only use policy sets with JAX-WS applications that run on the Axis2 web service engine. You cannot use policy sets for JAX-RPC applications.

About this task

You can use the administrative console to view and manage policy sets. From the administrative console, click **Services > Policy sets > Application policy sets** or **Services > Policy sets > System policy sets**. The Application policy sets collection displays a listing of the custom (if you have created custom policy sets) and default policy sets. Use the Application or System policy sets collection page to create, copy, delete, export, and import policy sets.

The following policy sets are ready for you to use as is.

- LTPA WSSecurity Default
- Kerberos V5 HTTPS default
- SSL WSTransaction
- Username SecureConversation
- Username WSSecurity default
- WS-Addressing default
- WSHTTPS default
- WS-I RSP ND
- WS-ReliableMessaging persistent

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

- “Viewing policy sets using the administrative console” on page 2667.
  This topic describes the process of viewing and evaluating policy sets.
- “Creating policy sets using the administrative console” on page 2668.
  This topic describes two ways to create policy sets: creating new policy sets or copying and renaming policy set templates.
- “Modifying policy sets using the administrative console” on page 2678.
  This topic describes how to edit custom policy sets you have created.
- “Importing policy sets using the administrative console” on page 2675.

This topic describes how to import policy sets from the default repository or from a selected location.

- "Exporting policy sets using the administrative console" on page 2759.
  This topic describes how to export policy sets.
- "Deleting policy sets using the administrative console" on page 2679.
  This topic describes how to delete custom policy sets. You can delete policy set templates and re-import them if needed.
- "Managing policies in a policy set using the administrative console" on page 2711
  This topic describes how you can define policies with policy sets to secure messages.
- "Defining and managing policy set bindings" on page 2680
  This topic describes configuring custom binding configurations.

Results

Using these tasks, you can determine how to create a new policy set and verify whether you can reuse an existing policy set. You can configure a policy set, and define policies for that policy set.

What to do next

Depending on how you are using policy sets, you might want to revisit some of the tasks listed in this topic to tweak the configuration for your policy set. You can also proceed to configure bindings for your policy set. See Defining binding information for policy sets.

Viewing policy sets using the administrative console

You can use the administrative console to view lists of policy sets. Policy sets can either be default policy sets that you cannot edit or custom policy sets that you have created and can edit. You can use policy sets, or assertions that define services, to simplify your web services configuration because policy sets group security and other web services settings into reusable units.

Before you begin

If you are creating a custom policy set by copying an existing default policy set, you might want to view the existing policy sets to choose a policy set with properties similar to the one you are planning to create.

About this task

You can view a list of policy sets to decide which policy sets can be reused or copied, modified and reused.

Procedure

1. To view policy sets from the administrative console, click **Services > Policy sets > Application policy sets** or **Services > Policy sets > System policy sets**.
2. Set the number of policy set rows that you want to view at a time using the Preferences settings. The default value is 20. The Editable column on the table in this view shows which policy sets you can edit. You can only edit the custom policy sets that you create. You cannot edit the provided default policy sets.
3. To view the details for any of the policy sets, click the name of the policy set in the Name column of the table. The Policy set settings page displays details about the selected policy set. If the policy set is a custom policy set you can edit the fields on the page. If the policy set is a provided default, then you can copy and reuse the policy set. You cannot edit the fields otherwise.

Results

After you have viewed the available policy sets and their settings, you can then decide if you want to create a new policy set, copy an existing policy set that you can rename, or use one of the existing policy sets that meets your needs.

Creating policy sets using the administrative console

You can use the administrative console to either create a policy set by specifying all the necessary information or by copying an existing policy set that you rename. You can use policy sets, or assertions that define services, to simplify your web services configuration because policy sets group security and other web services settings into reusable units.

Before you begin

To create a new policy set, you can either specify the information to create a new policy set or you can copy and rename an existing policy set. Using either method, you need basic information about the policy set that you want to create, such as the name, description, policies to include, policy details, attachments, and binding configurations. If you are creating a policy set by copying an existing policy set, then you should also view the existing policy sets to choose one with properties that are most similar to the one you plan to create.

About this task

Whether you choose to create a new policy set or copy and rename an existing one, start from the Applications policy sets collection in the administrative console.

Procedure

1. From the administrative console, click **Services > Policy sets > Application policy sets** or **Services > Policy sets > System policy sets**.
2. If the policy set you are creating is:
   - a new policy set, then click **New**.
   - an existing policy set to be copied and renamed, click the **Select** box beside the name of the policy set to be copied in the **Name** column and click **Copy**.

   Using either method, this action opens the Policy set settings view to specify the required information about the policy set being created or copied.
3. Enter the name of the policy set that you want to create or copy in the **Name** field.
4. Enter a brief description of the policy set in the **Description** field. This is the description that displays in the Application policy sets or the System policy set collection, so it must be meaningful to you and other potential users of this policy set.

   Note: If you created a new policy set, it does not contain policies to edit until you add them to the policy set. The policy set is initially empty.

Results

You have provided the basic information to create a policy set.

Example

After you have looked at your web services, you might decide that the WS-I RSP default policy set most closely meets your needs. You would go to the administrative console and click **Services > Policy sets > Application policy sets** to access the Application policy sets collection. Locate the WS-I RSP default in the **Name** column of the table and click the box beside it (in the **Select** column). Click **Copy**. This opens

the Policy set settings window. You might want to name your policy set by your company or division so you could provide a name like `ABC WS-I RSP` in the Name field. Because you know others in your organization might access and use it, you've chosen a name that is meaningful to those people too. You want to be sure everyone knows exactly what this copy of the WS-I RSP policy is used for, so you add a description in the **Description** field describing it. Now you want to customize the policy set so you edit the policy information by clicking the name of a policy to edit it.

When you identify the requirements of your web service, you might decide that none of the default policy sets meet your needs closely enough to use them as a template so you might decide to create your own policy set. You would first create the policy set with the name you choose to give it. As if you were reusing an existing template, you would go to the administrative console and click **Services > Policy sets > Application policy sets** and click **New**. The Policy set settings window opens but note that the Policy set name field is blank and there are not yet any associated policies in the table. Enter the name and add any policies necessary.

When you add policies to a policy set, the policies are set to their default values. You can then edit the policies to modify any attribute values that need to be changed and save the settings.

What to do next

If you are creating a new policy set without copying an existing policy set, you need to specify the policy information. If you are copying an existing policy set, you can either accept the default policies associated with the policy set or you can change the policies.

WS-I RSP default policy sets

The Reliable Asynchronous Message Profile (WS-I RSP) default policy sets are based on the Reliable Asynchronous Message Profile specification. The WS-I RSP default policy sets include the WS-I RSP default policy set, the Lightweight Third-Party Authentication (LTPA) WS-I RSP default policy set and the Username WS-I RSP default policy set. You can use these policy sets to simplify your web services configuration.

The WS-I RSP default policy sets are composed of a set of policies to provide reliable and secure web services. The WS-I RSP default policy sets use the WS-Addressing, WS-ReliableMessaging, and WS-Security specifications. Use the WS-I RSP default policy set, the LTPA WS-I RSP default policy set, or the Username WS-Security WS-I RSP default policy set as provided with the application server. To customize the policy sets, you must first copy the policy set, and then configure custom policy settings and bindings to meet your needs.

The WS-I RSP default policy sets include the following policies:

WS-Addressing policy
: You can use the WS-Addressing policy to enable the addressing capability of the WS-Addressing specification.

WS-ReliableMessaging policy
: You can use the WS-ReliableMessaging policy to specify the quality of service for reliable delivery.

WS-Security policy
: The WS-Security policy in the WS-I RSP default policy set provides the following security:

- Message integrity through digital signature that includes signing the body, time stamp, WS-Addressing headers and WS-ReliableMessaging headers using the WS-SecureConversation and WS-Security specifications.
- Confidentiality through encryption that includes encrypting the body, signature elements, using the WS-SecureConversation and WS-Security specifications.
- Traditional RSA cryptography is used to secure a request to a Trust Server to obtain a Secure Context Token (SCT). Thereafter, the conversation is secured using symmetric keys derived from the SCT.

The application server provides additional policy sets that you can use or customize. To use the following default policy sets, you must import the policy sets from the default repository. Read about importing policy sets using the administrative console for more information.

The following WS-I RSP default policy sets exist:

WS-I RSP default
This policy set provides:

- Reliable message delivery to the intended receiver by enabling WS-ReliableMessaging.
- Message integrity through digital signature that includes signing the body, time stamp, WS-Addressing headers and WS-ReliableMessaging headers using the WS-SecureConversation and WS-Security specifications.
- Confidentiality through encryption that includes encrypting the body, signature elements, using the WS-SecureConversation and WS-Security specifications.

LTPA WS-I RSP default
This policy set provides the WS-I RSP default policy set and adds a Lightweight Third Party Authentication (LTPA) token included in the request message to authenticate the client to the service.

Username WS-I RSP default
This policy set provides the WS-I RSP default policy set and adds a username token included in the request message to authenticate the client to the service. The username token is encrypted in the request.

SecureConversation default policy sets

The SecureConversation default policy sets are based on the Web Services Secure Conversation Language (SecureConversation) standard that establishes a secure context, based on shared keys for the client and server to use for a series of messages. This standard provides a framework to define how to secure the message exchange across organizations. The SecureConversation default policy sets include the SecureConversation policy set, the Lightweight Third-Party Authentication (LTPA) SecureConversation policy set, and the Username SecureConversation policy set.

The SecureConversation default policy sets are based on the WS-SecureConversation, the WS-Security, and the WS-Addressing specifications. Use the SecureConversation policy set, the LTPA SecureConversation policy set, or the Username SecureConversation policy set as provided with the application server. To customize the policy sets, you must first copy the policy set, and then configure custom policy settings and bindings to meet your needs.

The WS-SecureConversation specification alone does not provide a complete security solution. The WS-SecureConversation is built on the WS-Security and WS-Trust specifications to provide secure communication across one or more messages. Specifically, this specification defines mechanisms for establishing and sharing security contexts, and deriving keys from established security contexts or any shared secret.

WS-Security focuses on the message authentication model but not in a security context. The WS-SecureConversation specification defines mechanisms for establishing and sharing security contexts, and deriving keys from security contexts, to enable a secure conversation. By using the SOAP extensibility model, modular SOAP-based specifications are designed to be composed with each other to provide a rich messaging environment.

The following SecureConversation default policy sets exist:

SecureConversation
This policy set provides:

- Message integrity by digital signature that includes signing the body, timestamp, and WS-Addressing headers using WS-SecureConversation and WS-Security specifications.
- Message confidentiality by encryption that includes encrypting the body, signature and signature confirmation elements, using WS-SecureConversation and WS-Security specifications.

LTPA SecureConversation
This policy set provides the SecureConversation policy set and adds a Lightweight Third Party Authentication (LTPA) token included in the request message to authenticate the client to the service.

Username SecureConversation
This policy set provides the SecureConversation policy set and adds a username token included in the request message to authenticate the client to the service. The username token is encrypted in the request

WS-ReliableMessaging default policy sets

The WS-ReliableMessaging default policy sets are pre-configured to provide reliable message exchange between web services. Two of these policy sets (WS-I RSP and WS-I RSP ND) are immediately available, and the rest are readily available for import from a default repository.

With WS-ReliableMessaging, you can make your SOAP over HTTP-based web services reliable without writing custom code. You can use the provided non-editable default policy sets without change, or you can create customized copies of them.

All the default policy sets that include the WS-ReliableMessaging policy also include the WS-Addressing policy. The WS-ReliableMessaging policy provides the ability to deliver a message reliably to its intended receiver. The WS-Addressing policy provides a transport-neutral way to uniformly address web services and messages, and WS-ReliableMessaging uses WS-Addressing to provide asynchronous request and reply capabilities.

Note: WS-ReliableMessaging Version 1.1 messaging requires WS-Addressing to be mandatory. If you use a policy set that includes WS-ReliableMessaging and WS-Addressing policies, and the WS-Addressing policy is configured as optional, then WebSphere Application Server overrides the WS-Addressing setting and automatically enables WS-Addressing.

The following default policy sets that include the WS-ReliableMessaging policy are immediately available, as described in “Viewing policy sets using the administrative console” on page 2667:

WS-I RSP
This policy set enables WS-ReliableMessaging Version 1.1 and uses the minimum quality of service, *unmanaged non-persistent*. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages. In-order delivery is set to “false”, so messages are not necessarily delivered in the order in which they were sent. Message integrity is provided by digitally signing the body, the time stamp, and the WS-Addressing headers. Message confidentiality is provided by encrypting the body and the signature. This policy set follows the WS-SecureConversation and WS-Security specifications.

WS-I RSP ND
This is the network deployment version of the WS-I RSP policy set. This policy set provides the WS-I RSP default policy set and adds a *managed non-persistent* quality of service. This in-memory quality of service option uses a messaging engine to manage the sequence state, and messages are written to disk if memory is low. This quality of service allows for the re-sending of messages that are lost in the network, and can also recover from server failure. However, state is discarded after a messaging engine restart so in this case you will lose messages.

The following additional default policy sets that include the WS-ReliableMessaging policy are readily available for import, as described in “Importing policy sets using the administrative console” on page 2675:

LTPA WS-I RSP

This policy set provides the WS-I RSP default policy set and adds a Lightweight Third Party Authentication (LTPA) token included in the request message to authenticate the client to the service.

Username WS-I RSP

This policy set provides the WS-I RSP default policy set and adds a username token included in the request message to authenticate the client to the service. The username token is encrypted in the request.

WSReliableMessaging 1_0

This policy set enables both WS-ReliableMessaging Version 1.0 and WS-Addressing and uses the minimum quality of service, *unmanaged non-persistent*. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages. In-order delivery is set to “false”, so messages are not necessarily delivered in the order in which they were sent.

You can use this policy set with .NET-based web services.

WSReliableMessaging default

This policy set enables both WS-ReliableMessaging Version 1.1 and WS-Addressing and uses the minimum quality of service, *unmanaged non-persistent*. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages. In-order delivery is set to “false”, so messages are not necessarily delivered in the order in which they were sent.

WSReliableMessaging persistent

This policy set enables both WS-ReliableMessaging and WS-Addressing and uses the maximum quality of service, *managed persistent*. This quality of service supports asynchronous web service invocations and uses a service integration messaging engine and message store to manage the sequence state. Messages are processed within transactions, are persisted at the web service requester server and at the web service provider server, and are recoverable in the event of server failure. In-order delivery is set to “false”, so messages are not necessarily delivered in the order in which they were sent.

Because this policy set specifies managed persistent quality of service, you have to define bindings to the service integration bus and messaging engine that you want to use to manage the WS-ReliableMessaging state. You can attach and bind a WS-ReliableMessaging policy set to a web service application by using the administrative console or the wsadmin tool.

WSAddressing default policy set

The WSAddressing default policy set provides a transport-neutral way to uniformly address web services and messages.

The WSAddressing default policy set is based on the WS-Addressing specification. The WS-Addressing standard uses endpoint references and message addressing properties to facilitate the addressing of web services in a standard and interoperable way.

Use the WSAddressing default policy set as provided with the application server. To customize the policy set, you must first copy the policy set, and then configure custom policy settings and bindings to meet your needs.

To learn more about the WS-Addressing standard, read about Web Services Addressing support.

Web Services Security default policy sets

The Web Services Security default policy sets are based on the WS-Security 1.0 and Web Services Addressing (WS-Addressing) specifications. The Web Services Security default policy sets include the WSSecurity default policy set, the Lightweight Third-Party Authentication (LTPA) WSSecurity policy set, the Username WSSecurity policy set, and the Kerberos V5 HTTPS default policy set. These default policy sets are used to build secure web services.

The Web Services Security default policy sets use the WS-Security 1.0 specification enhancements to SOAP messaging to provide quality of protection through message integrity, message confidentiality, and single message authentication. Providing quality of protection means to prevent the following potential threats to SOAP messages:

- The message being modified or read by antagonists.
- An antagonist sending messages to a service that are formed correctly, but lack the appropriate security claims to be processed.

The WS-Addressing specification defines XML 1.0 and XML Namespaces elements to identify web services endpoints and to secure end-to-end endpoint identification in messages.

You can use the WSSecurity default policy set, the LTPA WSSecurity policy set, the Username WSSecurity policy set, or the Kerberos V5 HTTPS default policy set as provided with the application server. To customize the other Web Services Security policy sets, you must first copy the policy set, and then configure custom policy settings and bindings to meet your needs.

Features and details of the default Web Services Security policy sets are as follows:

Kerberos V5 HTTPS default

This policy set provides message authentication with a Kerberos Version 5 token. Message integrity and confidentiality are provided by Secure Sockets Layer (SSL) transport security. This policy set follows the OASIS Kerberos Token Profile V1.1 and WS-Security specifications.

When you use this policy set, configure the basic authentication data and custom properties such as the com.ibm.wsspi.wssecurity.krbtoken.targetServiceName and com.ibm.wsspi.wssecurity.krbtoken.targetServiceHost custom properties in the client bindings. For more information, see the Authentication generator or consumer token settings and Protection token settings (generator or consumer) topics.

LTPA WSSecurity default

This policy set provides:

- Message integrity through digital signature (using RSA public-key cryptography) to sign the body, time stamp, and WS-Addressing headers using WS-Security specifications.
- Message confidentiality through encryption (using RSA public-key cryptography) to encrypt the body, signature and signature elements using WS-Security specifications.
- A Lightweight Third Party Authentication (LTPA) token included in the request message to authenticate the client to the service.

Username SecureConversation

This policy set provides:

- Message integrity through digital signature that includes signing the body, time stamp, and WS-Addressing headers using WS-SecureConversation and WS-Security specifications
- Message confidentiality through encryption that includes encrypting the body, signature and signature confirmation elements, using WS-SecureConversation and WS-Security specifications
- A username token included in the request message to authenticate the client to the service. The username token is encrypted in the request

Username WSSecurity default

This policy set provides:

- Message integrity through digital signature (using RSA public-key cryptography) to sign the body, time stamp, and WS-Addressing headers using WS-Security specifications.
- Message confidentiality through encryption (using RSA public-key cryptography) to encrypt the body, signature and signature elements using WS-Security specifications.
- A username token included in the request message to authenticate the client to the service. The username token is encrypted in the request.

WSSecurity default

This policy set provides:

- Message integrity through digital signature (using RSA public-key cryptography) to sign the body, time stamp, and WS-Addressing headers using WS-Security specifications.
- Message confidentiality through encryption (using RSA public-key cryptography) to encrypt the body, signature and signature elements using WS-Security specifications.

WSTransaction default policy sets

The WSTransaction default policy sets are based on the WS-Transaction specification and provide transactional integrity for web services. The WSTransaction default policy sets include the WSTransaction policy set and the SSL WSTransaction policy set.

You can use the WSTransaction default policy sets to make your SOAP over HTTP-based web services interoperable and coordinate atomic transactions or business activities without writing custom code. Use the WSTransaction policy set or the SSL WSTransaction policy set as provided with the application server. To customize the policy sets, you must first copy the policy set, and then configure custom policy settings and bindings to meet your needs.

The WSTransaction default policy sets are:

WSTransaction

Use this policy set to coordinate distributed transactional work atomically and interoperably, by using the WS-AtomicTransaction specification. Also, use this policy set to coordinate loosely coupled business processes that are distributed across the heterogenous web service environment, with the ability to compensate actions if a failure occurs in the business activity, by using the WS-BusinessActivity specification.

SSL WSTransaction

Use this policy set to coordinate distributed transactional work atomically, interoperably and securely, by using the WS-AtomicTransaction specification and SSL Transport security. Also, use this policy set to coordinate loosely coupled business processes, with the ability to compensate actions if a failure occurs in the business activity, securely, by using the WS-BusinessActivity specification and SSL Transport security.

WSHTTPS default policy set

The WSHTTPS default policy set provides SSL transport security for the HTTP protocol with web services applications.

The WSHTTPS default policy set is provided with the application server and it contains the HTTP transport policy, the SSL transport policy and the WS-Addressing policy.

You can use the WSHTTPS default policy set as provided with the application server. You cannot edit the WSHTTPS default policy set. You can create a copy of the default policy set and then configure custom policy settings and bindings to meet your needs. Alternatively, you can create a new policy set and specify the policies for it.

Copy of default policy set and bindings settings

Use this page to copy a policy set that you select from a list of available policy sets.

To view this administrative console page:

1. Click **Services > Policy sets > Application policy sets**.
2. Select the policy set that you want to copy, and click **Copy**.
3. Enter a name for the copy of the policy set in the **Name** field.
4. [Optional] Enter a description for the copy of the policy set in the **Description** field.

Name:

Specifies the name of the policy set. Use this field to enter a name for the copy of the policy set.

Description:

Specifies a description of the policy set that you want to copy.

Transfer attachments:

If the policy set that you want to copy is attached to one or more applications, services, or endpoints, then the check box option for **Attach this policy set in place of the original for all attached applications, services, endpoints, and operations**, is available. The default setting for this check box is cleared. You can perform the following actions:

1. Select the **Attach this policy set in place of the original for all attached applications, services, endpoints, and operations** check box to move the attachments to a new policy set. Selecting the check box detaches the original policy set and attaches the replacement policy set in its place.
2. Select **Copy bindings** if you want to copy the bindings of the policy set that is currently attached.
3. [Optional] Select **Restore default bindings** if you want to restore the default bindings.

Importing policy sets using the administrative console

You can import predefined policy sets or import policy sets from a selected location using the administrative console.

Before you begin

Before you begin this task, review Application policy sets collection.

About this task

The policy sets panel has an Import button to allow the import of policy sets. You can import a policy set from the default repository or from a selected location.

Procedure

1. From the administrative console navigation, click **Services > Policy sets > Application policy sets** or **Services > Policy sets > System policy sets**. The policy sets collection page displays a listing of the custom and default policy sets. Custom policy sets are displayed only if you have created them. If you have not created a custom policy set, then only the default policy sets are displayed.
2. From the Application policy sets panel, click **Import**.
3. Choose the option for importing the policy set. You can choose to import a policy set from the default repository or from a selected location. Selecting **Import from Default Repository** accesses the next panel that displays a set of predefined default policy sets that are available for you to import. Some of the predefined policy sets have already been imported for you from the default repository. You can import additional predefined policy sets that are available in the predefined policy set repository.

   Select **Import from Selected Location** to provide a path or browse your file system for a policy set file to import.

   - To import a policy set from the default repository, select **Import from Default Repository** and perform task step 4.

- To import a policy set from a selected location, select **Import from Selected Location** and perform task step 7.

4. Select **Import from Default Repository** and select either the **Import** or the **Import a Copy** radio button.

   The behavior of **Import** policy set:

   - Available for selection of one or more policy sets.
   - Imports the default, not editable, policy set. You are returned to the Application policy sets panel after the import of the policy set.
   - The imported policy set can be deleted, but it cannot be modified.

   The behavior of **Import a Copy** for a policy set:

   - Not available for multiple selection. If multiple policy sets are selected and the **Import a Copy** radio button is selected, an error message is displayed, indicating that only one policy set can be copied at a time.
   - Prompts you for a name and description for the copy. You are returned to the Application policy sets panel after you import the policy set.
   - The imported policy set can be modified or deleted.

5. If you selected **Import** from the previous step, select one or more policy sets to import in the **Name** column.
   a. Click **OK**. You are returned to the Application policy set panel.
   b. Click **Save**, to save your changes to the configuration.
6. If you selected **Import a Copy** from step 4, select the policy set to import a copy in the **Name** column. You can only select one policy set in the Name column to import its copy.
   a. Provide a different name for the copy.
   b. Click **OK**. You are returned to the Application policy set panel.
   c. Click **Save**, to save your changes to the configuration.
7. Select **Import from Selected Location**.
   a. Specify the full path or browse for the policy set file in your file system.
   b. Select either **Use current policy set name** or **Specify a different name to use for this policy set**.
   c. Click **OK**. You are returned to the Application policy set panel.
   d. Click **Save** to save your changes to the configuration.

Results

When you finish this task, you have been able to import a policy set using either the default repository or from a selected location.

Example

If you have a policy set, `XYZ_ps` and you want to import it from a selected location, perform the following steps:

1. From the Application policy set panel, click **Import**.
2. Select **Import from Selected Location**.
3. Specify the full path or browse to where the `XYZ_ps` policy set file is located in your file system.
4. Select either **Use the current policy set name**, `XYZ_ps`, or **Specify a different name for the policy set**.
5. Click **OK**.
6. Click **Save**, to save your changes to the configuration.

What to do next

You can modify or use the policy set that you just imported.

Import policy sets from default repository settings

Use this page to specify a policy set to import from a default repository. You can import predefined policy sets from the default repository.

To view this administrative console page, complete the following actions:

1. Click **Services** > **Policy sets** > **Application policy sets**.
2. Click **Import**.
3. Select **From Default Repository**.
4. You have the Import and Import a copy options. Select **Import a copy**.
5. In the **Name of copy** field, enter a name for the copy of the policy set that you select to import from the list of policy sets.

The behavior of **Import**:

- Available for selection of one or more policy sets.
- Imports the default, not editable, policy set. You are returned to the Application policy sets panel after the import of the policy set.
- The imported policy set can be deleted, but it cannot be modified.

The behavior of **Import a copy** policy set:

- Not available for multiple selection. If multiple policy sets are selected and the **Import a copy** radio button is selected, an error message is displayed, indicating that only one policy set can be copied at a time.
- Prompts you for a name for the copy. The description of the imported policy set is kept, but you can edit the description after you have imported the policy set. You are returned to the Application policy sets panel after the import of the policy set.
- The imported policy set can be modified or deleted.

Name of copy:

Use this field to enter a name for the copy of the policy set that you select to import from the list.

Before you can edit this field, you must select **Import a copy** option.

Buttons:

Click the appropriate button:

Button	Resulting Action
OK	Imports the specified policy set. You are returned to the Application policy set panel after the import.
Cancel	Returns you to the Application policy set panel without importing a policy set.

Import policy sets from a selected location settings

Use this page to specify a policy set to import from a selected location.

To view this administrative console page, click **Services > Policy sets > Application policy sets > Import policy set from selected location**.

On this page, specify a policy set to import from a selected location.

Path and file name for the policy set file:

Specify a fully qualified path and file name to the policy set file or use the **Browse** to locate the file in your file system.

Select **Use current policy set name** if you are not changing the policy set name and click **OK**.

Select **Specify a different name for this policy set** if you are changing the policy set name. Provide a new name and click **OK**.

Button	Resulting action
OK	Imports the specified policy set file if valid or returns an error if an invalid file is provided. You are returned to the Application policy set panel after the import.
Cancel	Returns you to the Application policy set panel without importing a policy set.

Modifying policy sets using the administrative console

You can use the administrative console to modify existing custom policy sets that you have created. If you have copied an existing default policy set or created a policy set yourself, you can always go back and make changes to them to make them better suit the changing needs of your business.

Before you begin

You can create a copy of a policy set and modify the copy. You must first copy a default policy set before you can modify it. See copy of default policy set and bindings settings.

About this task

To modify a custom policy set, you can use the Policy set settings view to access configuration information for the policy set.

Procedure

1. Create a policy set using the steps in “Creating policy sets using the administrative console” on page 2668
2. From the administrative console, click **Services > Policy sets > Application policy sets** >*policy_set_name* or **Services > Policy sets > System policy sets** >*policy_set_name* where *policy_set_name* is the custom policy set of interest that is displayed in the table. This opens the Policy set settings view. You must specify the required information about the policy set that you want to modify.
3. Change the description of the policy set. Edit the brief description of the policy set in the Description field. This is the description that displays in the Application policy sets collection so it must be meaningful to you and other potential users of this policy set.
4. Edit the policy information. You can include, exclude, add or delete policies from this policy set using the settings in the Policies table. To work more extensively with policies, see “Managing policies in a policy set using the administrative console” on page 2711.
5. This step does not apply to system policy sets. Detach the policy set from an application or replace the policy set attached to the application with another policy set. Click the **Attached applications** link to access applications that are attached to this policy set and applications that contain attached service resources. To detach a policy set from an application, use the following steps:
   a. In the Filter list, click **Name**.
   b. Find the application. In the Search term(s) field, enter all or part of the name of the application you want to detach and click **Go**.
   c. Detach the application. Find and select the application in the listing and click **Detach Policy Set**.

To change the policy set that is attached to an application, use the following steps:

a. In the Filter list, click **Name**.
b. Find the application. In the Search term(s) field, enter all or part of the name of the application and click **Go**.
c. Replace the policy set for this application. Find and select the application in the listing and click **Replace Policy Set**. You then have one of the following options:
   - If there are fewer than 25 policy sets to choose from, the Replace Policy Set button is a drop down list and you can click a policy set to attach to the application.
   - If there are more than 25 policy sets to choose from, the Replace Policy Set button opens a collection of policy sets and you can select the policy set to replace from the listing.

Results

Performing these tasks modifies the application to which the custom policy set is attached.

Example

If you had a custom policy set ABC WS-I RSP, for example, that had a misleading description and you want to change the description so that other users know what the policy set really includes, you would access the Policy Set settings view for that policy set. To do this, from the administrative console, you would click **Services > Policy sets > Application policy sets** > ***ABC WS-I RSP***. Then you would click the description field and edit the text so that it better represents the scope of the policy set.

What to do next

If the policy set you have modified is an attached policy set, restart all affected applications to pick up the changes you made. If the policy set is unattached, then no further action is required.

Deleting policy sets using the administrative console

You can use the administrative console to delete the default policy sets or the application specific policy sets that you have created.

Before you begin

If you are deleting an attached policy set, stop all applications and the associated endpoints or operations, then detach and delete the policy set. If the policy set is unattached, then you do not need to detach the policy set before deleting it.

About this task

If a application specific policy set is no longer needed, you might choose to delete it. To delete a application specific policy set, use the Application policy set collection panel.

Procedure

1. From the administrative console, click **Services > Policy sets > Application policy sets** or **Services > Policy sets > System policy sets**.
2. Click the box in the **Select** column next to the policy set that you want to delete.
3. Click **Delete**.
4. Verify that you want to delete this policy set.

Results

The selected application specific policy set is deleted from the list of policy sets and is no longer available.

Example

You might decide that the *ABC WS-I RSP* policy set that you created did not meet your needs as you had expected and you have already created the *DEF WS-I RSP* policy set to better meet your needs. You have detached all the applications and endpoints from the *ABC WS-I RSP* policy set and you want to delete the policy set. Click **Services > Policy sets > Application policy sets** and find the *ABC WS-I RSP* policy set in the listing. Select the *ABC WS-I RSP* policy set and click **Delete**. Confirm that you want to delete the policy set and it is removed from the listing of policy sets. Restart the applications that you detached the ABC WS-I RSP policy from, and you have successfully deleted the policy set.

Alternatively, and depending on your use case, you might want to reassign the attachment to another resource before deleting the policy set. Make a copy of the policy set that you are about to delete, then attach it later to your application or service artifact. See attaching a policy set to a service artifact. You can now delete the policy set.

What to do next

If you have detached and then deleted a policy set from an application, restart all affected applications to pick up this change.

Defining and managing policy set bindings

Policy set bindings contain platform specific information, like keystore, authentication information or persistent information, required by a policy set attachment. Use this task to create and manage bindings.

About this task

In Version 7.0 and later, there are two types of bindings, application specific bindings and general bindings.

Application specific bindings

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and constrained to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have very limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service providers, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service provider resources that have an attached policy set. See service providers policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings were introduced in Version 7.0. These bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though

general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general provider policy set bindings by accessing **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. See defining and managing service client or provider bindings. General provider policy set bindings might also be used for trust service attachments.

Important: The general bindings that are shipped with the product are provider and client sample bindings. These bindings are initially set as the cell default bindings. Do not use these bindings in their current state in a production environment. To use the sample bindings, modify them to meet your security needs in a production environment. Alternatively, create a copy of the bindings and then modify the copy. For example, change the key and keystore settings to ensure security, and modify other settings to match your environment. You must also configure the username and password for Username or LTPA token authentication. See the topic Configuring the username and password for WS-Security Username or LTPA token authentication for more information.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

To view or work with your current policy sets bindings, perform the following:

Procedure

1. To view your current policy set and application specific bindings from the administrative console, click **Services > Policy sets > Application policy sets** >*policy_set_name* > **Attached applications** and then click an application.

   Depending on the application that you select, you can manage the bindings attached to the following policy sets:

   - Service provider policy sets and bindings
   - Service client policy sets and bindings

   To learn more about managing the bindings attached to policy sets, see the service provider or service client policy sets and bindings information.

   Sort on the **Attached policy set** column on either of the policy sets and bindings pages to select the service resources with the same policy set attached. Likewise, sort on the **Binding** column to select the service resources that share the same custom binding to attach to a different policy set. If you sort on the **Policy Set** or the **Binding** column, the hierarchical relationship of the service resources in the first column is not accurate. You can sort again on the **Application/Service/Endpoint/Operation** column to restore the hierarchical relationship. The entries in the **Application/Service/Endpoint/ Operation** column display in ascending order.
2. To work with an existing bindings from the administrative console, click **Services > Policy sets > Application policy sets** >*policy_set_name* > **Attached applications**. Click an application name, and then click either the **Service provider policy sets and bindings** or the **Service client policy sets and bindings**. Then click a binding name in the Bindings column of the table.

   Note: If no applications appear when you click **Attached applications**, you do not have any applications attached to the selected policy set. To attach a policy set and binding to an application using the administrative console, click **Applications > Enterprise Applications >**

application name. Then, click either **Service provider policy sets and bindings** or **Server client policy sets and bindings** to attach resources to your policy set and to assign bindings.

.

3. [Optional] To work with general bindings, click **Services > Policy sets > General client policy set bindings** or **Services > Policy sets > General provider policy set bindings**.

   You can complete the following actions for general bindings:

   - "Importing policy set bindings using the administrative console"
   - "Export policy sets bindings settings" on page 2689
   - "Defining and managing service client or provider bindings" on page 2685
   - "Creating new or configuring existing general binding settings" on page 2688
   - "Copy policy set binding settings" on page 2689
   - "Deleting policy set bindings" on page 2690

Results

When you finish this task, you would have performed one or more of the following:

- Created an application specific or general policy set binding
- Imported a policy set binding
- Exported a policy set binding
- Deleted a policy set binding
- Modified an application attachment to apply an application specific binding for single security domain
- Modified an application attachment to apply an application specific binding for multiple security domain

Importing policy set bindings using the administrative console

You can import predefined client or provider policy set bindings using the administrative console.

Before you begin

This task only applies to general client or provider bindings. Before you begin this task, review the defining and managing service client or provider bindings.

About this task

The general bindings panel has an Import button to allow the import of client or provider policy set bindings. You can import a policy set binding by specifying the full path and filename of the binding to import.

Procedure

1. Navigate to the General client policy set bindings or the General provider policy set bindings panel using one of the following ways:
   - **Services > Policy sets > General client policy set bindings**
   - **Services > Policy sets > General provider policy set bindings**
2. Click **Import**.
3. Specify the full path and file name of the policy set binding to import.
4. Select either **Use current binding name** or **Specify a different name to use for this binding**. If you are importing a binding and that binding name already exists in the repository, you must specify a different name to use for the binding import.
5. Click **OK**.

Results

When you finish this task, you have been able to import a policy set binding using the administrative console.

Example

If you have a policy set binding, `XYZ_psbind` and you want to import it, perform the following steps:

1. From the General client or provider policy set bindings panel, click **Import**.
2. Specify the full path and file name of the policy set binding to import or use **Browse** to locate the binding in your file system.
3. Select either **Use current binding name** or **Specify a different name to use for this binding**.
4. Click **OK**.
5. Click **Save**, to save your changes to the configuration.

What to do next

You can export a policy set binding to reuse it.

Import policy set bindings settings:

Use this page to specify a service client or provider policy set binding to import for your service.

Navigate to the General client policy set bindings or the General provider policy set bindings panel using one of the following ways:

- **Services > Policy sets > General client policy set bindings**
- **Services > Policy sets > General provider policy set bindings**

Click **Import**. Specify a fully qualified path and binding name that you want to use to import the policy set binding. The policy set binding file must be a compressed file.

Full Path with file name:

Specify a fully qualified path and file name to the policy set file or use the browse button to locate the file in your file system.

Select **Use current binding name** if you are not changing the binding name and click **OK**.

Select **Specify a different name for this binding** if you are changing the binding name. Provide a new name and click **OK**.

Button	Resulting action
OK	Imports the specified binding file if valid or returns an error if an incorrect file is provided. You are returned to the General client or provider policy set binding panel after the import.
Cancel	Returns you to the General client or provider policy set binding panel without importing a binding.

Web services policy set bindings

A set of bindings is a named object that is associated with a specific policy set and service resource that is attached to the policy set.

Bindings contain environment and platform specific information, like the following types of information:

- Keys used for signature and encryption
- Keystore information
- Authentication information
- Persistent information

Typically, bindings are specific to the application or the platform, and they are not shared.

There are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and defined by the characteristics of the policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service providers, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service provider resources that have an attached policy set. See service providers policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Although general bindings are highly reusable, they do not provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general provider policy set bindings by accessing **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. See defining and managing service client or provider bindings. General provider policy set bindings might also be used for trust service attachments.

Important:

The sample general bindings that are shipped with the product are provider and client sample. Do not use these sample bindings in their current state in a production environment. However, if they were modified to contain non-sample data, you can use these sample bindings in a production environment.

You cannot assign a binding to a service provider resource that does not have a policy set or has an inherited attachment. To assign a binding to such a service provider resource, you

must first attach a policy set to the resource. Also, you cannot assign a binding to a service client resource that does not have an effective policy configuration or has an inherited policy attachment. To assign a binding to such a service client resource, you must first attach a policy set or specify the use of the provider policy.

Defining and managing service client or provider bindings

Service client or provider bindings are general bindings. You can create, copy, and manage general bindings such as the client or provider policy set bindings. These bindings provide system-specific configuration and can be reused across policy set attachments.

Before you begin

You cannot assign a binding to a service provider resource that does not have a policy set or has an inherited attachment. To assign a binding to such a service provider resource, you must first attach a policy set to the resource. Also, you cannot assign a binding to a service client resource that does not have an effective policy configuration or has an inherited policy attachment. To assign a binding to such a service client resource, you must first attach a policy set or specify the use of the provider policy. For more information, read about attaching a policy set to a service artifact.

About this task

There are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and defined by the characteristics of the policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in an unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service providers, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service provider resources that have an attached policy set. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set.

General bindings

General bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they do not provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

Important: The general bindings that are included with the product are provider and client sample bindings. Do not use these bindings in their current state in a production environment. However, if they were modified to contain non-sample data, they could be used in a production environment.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

You can complete the following tasks to define and manage general client or provider policy set bindings.

Procedure

1. To create a new general client or provider policy set binding or to manage the binding configuration from the administrative console, click **Services > Policy sets > General client policy set bindings** > **New**. You can also access this panel by clicking **Services > Policy sets > General provider policy set bindings** > **New**. Use the resulting detail panel to create a new client or provider policy set binding. For more information, read about creating new or configuring existing general binding settings.
2. To copy a specific policy set binding, select the binding name from the table and click **Copy**. For more information, read about copying a policy set binding settings.
3. To import a client or provider policy set binding, click **Import**. Read about importing policy set bindings using the administrative console to complete the import task.
4. To export a client or provider policy set binding, select the binding name from the table, and click **Export**. For more information, read about export policy set binding settings.
5. To delete a policy set binding, select the binding name from the table, and click **Delete**. For more information, read about deleting policy set bindings.

Results

When you finish this task, you have created, copied, exported, imported or deleted a client or provider policy set binding.

Service client or provider policy set bindings collection:

Use this page to create, copy, and manage general policy set bindings, such as the service client or provider bindings. These bindings provide system-specific configuration, and can be reused across policy set attachments. You can select the general default bindings, create new general bindings, or use existing bindings for an attached policy set.

To view this administrative console page, click **Services** > **Policy Sets** > **General client policy set bindings**. You can also view this page by clicking **Services** > **Policy Sets** > **General provider policy set bindings**.

About policy set bindings

There are two types of bindings, application specific bindings and general bindings.

Application specific binding

You can create application specific bindings only at a policy set attachment point. These bindings are specific to and defined by the characteristics of the defined policy. Application specific bindings can provide configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have limited reuse across policy sets.

When you create an application specific binding for a policy set attachment, the binding begins in an unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application specific bindings.

For service providers, you can only create application specific bindings by selecting **Assign Binding** > **New Application Specific Binding** for service provider resources that have an attached policy set. See service providers policy sets and bindings collection. Similarly, for service clients, you can only create application specific bindings by selecting **Assign Binding** > **New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

General bindings were introduced in WebSphere Application Server Version 7.0. These bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general provider policy set bindings by accessing **Services** > **Policy sets** > **General provider policy set bindings** > **New** in the general provider policy sets panel. Or, you can create general client policy set bindings by accessing **Services** > **Policy sets** > **General client policy set bindings** > **New** in the general client policy sets panel. See defining and managing service client or provider bindings. General provider policy set bindings might also be used for trust service attachments.

Important:

The sample general bindings that are shipped with the product are provider and client sample. Do not use these sample bindings in their current state in a production environment. However, if they were modified to contain non-sample data, you can use these sample bindings in a production environment.

You cannot assign a binding to a service provider resource that does not have a policy set or has an inherited attachment. To assign a binding to such a service provider resource, you must first attach a policy set to the resource. Also, you cannot assign a binding to a service client resource that does not have an effective policy configuration or has an inherited policy attachment. To assign a binding to such a service client resource, you must first attach a policy set or specify the use of the provider policy.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Name:

Specifies the name of the service client or provider policy set binding.

The following list of buttons are available to create and manage service client or provider policy set bindings:

Button	Resulting Action
New	Creates a new client or provider policy set binding. This option accesses the New binding panel. The binding is empty initially; you must enter a name and an optional description for a new client or provider policy set binding.

Button	Resulting Action
Delete	Removes the selected client or provider general policy set binding. The default general bindings are being used as the default for a server or a security domain, including the global security domain. You cannot delete default bindings for global security or other security domains. An attempt to delete such default general binding generates an error message that the selected binding cannot be deleted because it is currently the default binding for a security domain.
Copy	Creates a modifiable copy of the selected client or provider policy set binding using a new name that you provide.
Import	Imports a client or provider policy set binding. This is a menu item with the option of importing a binding with the same name or providing a different name for the import.
Export	Exports the selected client or provider policy set binding to an archive file.

Security Domain:

Indicates which security domain uses the general policy set binding. The default domain is global security. This column in the collection table appears only when there are multiple security domains to which the binding can be scoped.

Description:

Specifies the user-defined description for the client or provider policy set bindings.

Creating new or configuring existing general binding settings:

Use this page to create a new client or provider policy set binding. You can also use this page to configure an existing general binding. Empty bindings will be deleted. Scoping a binding to a security domain constrains the configuration options to those applicable to that domain and limits use of the binding to the specified domain.

To view this administrative console page, click **Services > Policy sets > General client policy set bindings** > **New**. You can also view this page by clicking **Services > Policy sets > General provider policy set bindings** > **New**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Bindings configuration name:

Specifies the name of the new binding configuration.

You must enter a binding configuration name to proceed. The name of the binding is automatically applied when you add a policy type and provide additional configuration information. At that time, the initial **Cancel** button is replaced with an **OK** button. Each binding that you add is initially empty, and you are directed to the binding configuration panel to provide additional information for the binding.

Button	Resulting Action
Add	Adds a policy type, such as WS-Security, to a new binding that you create or to an existing binding. You can access the configuration of each policy type binding through the policy type name link. If you click **Cancel** without providing any configuration information for the policy type, that policy type is not saved or included in the list of policy types. You might receive an attention message that the policy type was not saved as part of the binding configuration.

Button	Resulting Action
Delete	Deletes a policy type from the list.
OK	Associates the name of the binding to the policy type information after you add a policy type and provide additional configuration information.
Cancel	Returns you to the General client or provider policy set bindings panel without creating a new binding configuration.

Security domain:

This field is only available in a multiple security domain environment and specifies a valid security domain. You can scope the binding to that particular security domain, which is scoped to the global security domain by default.

Description:

Specifies a brief description of the new client or provider policy set binding.

Export policy sets bindings settings

This task only applies to general client or provider bindings. Use this page to export either a client or provider policy set binding for reuse.

Navigate to the General client policy set bindings or the General provider policy set bindings panel using one of the following paths:

- **Services** > **Policy sets** > **General client policy set bindings**
- **Services** > **Policy sets** > **General provider policy set bindings**

1. Select a binding from the list and click **Export**.
2. Click the binding to download the archive file.
3. Use the file dialog box to select a location for the exported binding. The policy set binding file is exported as a compressed file.
4. Click **Done**.

Copy policy set binding settings

Use this page to view and copy general policy set bindings for either a single security or a multiple security domain environment.

To access this administrative console page, perform the following steps:

1. Click **Services** > **Policy sets** > **General client policy set bindings**. You can also access this page by clicking **Services** > **Policy sets** > **General provider policy set bindings**.
2. Select an existing binding from the list, and click **Copy**.
3. Provide a name and description for the copy.
4. Click **OK**.
5. Click **Save**, to save your changes.

Name:

Specifies the name of copy. You must provide a different name for the copy of the binding configuration.

Security domain:

Specifies a valid security domain. This field is only available in a multiple security domain environment. You can scope the binding to a particular security domain; it is scoped to the global security domain by default.

Description:

Specifies a brief description of the copy of the policy set binding.

Deleting policy set bindings

Use this task to delete general policy set bindings using the administrative console.

About this task

You cannot delete general bindings specified as the default for any server or domain. You also cannot delete general bindings that are assigned to a policy set attachment. To delete other bindings, perform the following steps:

Procedure

1. Click **Services > Policy sets > General client policy set bindings**. You can also access this page by clicking **Services > Policy sets > General provider policy set bindings**.
2. Select an existing binding from the list, and click **Delete**.
3. Click **Save**, to save your changes.

Results

When you finish this task, you have deleted a general policy set binding.

Creating application specific bindings for policy set attachment

After you attach a policy set to a service artifact such as an application, service, or endpoint, you can define application specific bindings for the attached policy set.

Before you begin

Before you can start this task, you must deploy an application containing web services, and attach a policy set. See attaching a policy set to a service artifact.

About this task

The application specific binding panels display options based on the definitions in the attached policy set. For example, if the policy set does not have WS-ReliableMessaging configured, then the application specific binding panels do not display an option to configure WS-ReliableMessaging bindings. To create application specific bindings for an attached policy set, perform the following steps:

Procedure

1. Open the administrative console.
2. To create application specific bindings for a service provider policy attachment, click **Applications > Application Types > WebSphere enterprise applications** >*Service_provider_application_instance* > **Service provider policy sets and bindings**.

   To create application specific bindings for a service client policy attachment, click **Applications > Enterprise applications** > ***application_name*** > **Service client policy sets and bindings**.
3. Select the check box for the service artifact with an attached policy set.
4. Click **Assign Binding** to see a list of available bindings to assign.
5. [Optional] Select a previously defined application specific attachment binding from the list.

   If you decide to create a new application specific binding, complete steps 6 through 9.
6. Click **New**.
7. Specify a name for the binding and select which policies that you want to include in this application specific binding. Click **Add**, and select the policies to add.

8. Complete the information for the specified policies to define associated configuration data.
9. Click **Save**, to save your changes to the master configuration.

Results

When you finish this task, a application specific binding is created for the service artifact with the attached policy set.

Example

You have the application, `app1` with the attached policy set `Username WSSecurity default`, and you want to define application specific bindings. First locate the application in the **Applications > Application Types > WebSphere enterprise applications** collection. Click the **app1** application. You can then click the **Service provider policy sets and bindings** link or the **Service client policy sets and bindings** link. Select the check box for the `app1` application where the policy set is attached. Click **Assign Binding** and click **New**. Specify a name for the binding, such as `myBinding`. Click **Add** and select **WS-Security** from the list. Next, click the link for **WS-Security** policy and complete the panels with the appropriate information. Click **Save**, to save your changes to the master configuration.

What to do next

Restart the application that contains the service artifact with the application specific bindings you created.

Setting server default bindings for policy sets

You can set server default bindings if you want the policy set attachments for service providers and clients that are deployed to the server to use bindings that are different than those that are specified for the cell. If you use multiple security domains, your default server bindings will also override the security domain default bindings.

Before you begin

Before you can set server default bindings for your Java API for XML-Based Web Services (JAX-WS) application, you must first configure at least one general provider policy set binding or general client policy set binding. To define and manage these general bindings, use the administrative console and select **Services > Policy sets > General provider policy set bindings** or **Services > Policy sets > General client policy set bindings** .

About this task

Note: In WebSphere Application Server Version 7.0 and later, the security model was enhanced to a domain-centric security model instead of a server-based security model. The configuration of the default global security (cell) level and default server level bindings has also changed in this version of the product. In the WebSphere Application Server Version 6.1 Feature Pack for Web Services, you can configure one set of default bindings for the cell and optionally configure one set of default bindings for each server. In Version 7.0 and later, you can configure one or more general service provider bindings and one or more general service client bindings. After you have configured general bindings, you can specify which of these bindings is the global default binding. You can also optionally specify general binding that are used as the default for an application server or a security domain.

General service provider and client bindings are not linked to a particular policy set, and they provide configuration information that you can reuse across multiple applications. You can create and manage general provider and client policy set bindings, and then select one of each binding type to use as the default for an application server. Setting the server default bindings is useful if you want the services that are deployed to a server to share binding configuration. You can also accomplish this sharing of binding

configuration by assigning the binding to each application deployed to the server or by setting default bindings for a security domain and assigning the security domain to one or more servers.

You can specify default bindings for your service provider or client that are used at the global security (cell) level, for a security domain, for a particular server. The default bindings are used in the absence of an overriding binding specified at a lower scope. The following list is the order of precedence from lowest to highest that the application server uses to determine which default bindings to use:

1. Server level default
2. Security domain level default
3. Global security (cell) default

The sample general bindings that are provided with the product are initially set as the global security (cell) default bindings. The default service provider binding and the default service client bindings are used when no application specific bindings or trust service bindings are assigned to a policy set attachment. For trust service attachments, the default bindings are used when no trust specific bindings are assigned. If you do not want to use the provided **Provider sample** as the default service provider binding, you can select an existing general provider binding or create a new general provider binding to meet your business needs. Likewise, if you do not want to use the provided **Client sample** as the default service client binding, you can select an existing general client binding or create a new general client binding. To specify your global security (cell) default bindings, use the administrative console, and click **Services > Policy sets > Default policy set bindings**. For environments with multiple security domains, you can optionally choose the general provider and general client bindings that you want to use as the default bindings for a domain.

In addition to choosing default bindings for the global security (cell), you can also choose the general provider and general client bindings that you want to use as the default bindings for a server. This is only necessary if you want to use different default bindings for a particular server than those used by the other servers in the security domain or cell.

To choose the default bindings for a server from the administrative console, click **Servers > Server Types > WebSphere application servers > *server_name*** and then under Security, click **Default policy set bindings**. If you do not choose a general binding as the default for a server, the default bindings for the domain in which the server resides is used. If you do not choose a binding as the default for a domain, the default bindings for the global security (cell) are used. You must choose a default service provider and default service client bindings for the cell. The general bindings that are included with the product are initially set as the global security (cell) default bindings. You cannot delete a binding that is used as part of any policy set attachment or specified as the default binding for server, a domain, or the cell. To learn more about defining default bindings for a server, see the server default bindings documentation.

Note:

> If you have an application that contains one or more application specific bindings that are configured at the WebSphere Application Server Version 6.1 level, this application is a V6.1 application. If you have applications that are deployed to V6.1 servers within the Version 7.0 or later application server environment, or you have V6.1 applications that are deployed to V7.0 or later versions of the application server, you can specify Version 6.1 default policy set bindings for the cell. These bindings are used for both client and provider policy set attachments within V6.1 applications and attachments to service applications that are deployed to a V6.1 server. Additionally, these default bindings are used for V6.1 attachments unless they are overridden at the attachment point by an application specific binding or a V6.1 server default binding. You can upgrade V6.1 bindings to the bindings that are used by WebSphere Application Server V7.0 and later versions. Use the upgradeBindings command using the wsadmin tool to upgrade the bindings level, if the V6.1 application is not installed on WebSphere Application Server V6.1.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation

to learn more about the valid roles for the application server.

Procedure

1. Open the administrative console.
2. To set default policy set bindings for your server, select**Servers > Server Types > WebSphere application servers > *server_name* > Default policy set bindings**.
3. Select the server default provider binding.

   If you specify a server default provider binding, the selected binding overrides the default provider bindings that are specified for the cell or the security domain to which the server is deployed. The default setting is **None**.

   If multiple security domains are in use, the name of the security domain to which each binding is scoped displays beside the name of each available provider binding. Only the bindings that are scoped to the global security level or to the security domain to which the server is deployed are displayed.
4. Select the server default client binding.

   If you specify a server default client binding, the selected binding overrides the default client bindings that are specified for the cell or the security domain to which the server is deployed. The default setting is **None**.

   If multiple security domains are in use, the name of the security domain to which each binding is scoped displays beside the name of each available provider binding. Only the bindings that are scoped to the global security level or to the security domain to which the server is deployed are displayed.
5. Click **Apply** or **OK** to submit your changes.
6. Click **Save** to save your changes to the master configuration.
7. (optional) If you are using a Version 6.1 application, you can specify server V6.1 default policy set bindings. To set these bindings, select **Servers > Server Types > WebSphere application servers > *server_name* > Default policy set bindings > Version 6.1 default policy set bindings**.

   Note: Select the V6.1 default bindings for this server. If your application contains one or more application specific bindings that are configured at the WebSphere Application Server V6.1 level, this application is a V6.1 application. These default bindings are used for client and provider policy set attachments for applications that are deployed to V6.1 servers, and for V6.1 applications that are deployed to V7.0 and later servers. These bindings are used for both client and provider policy set attachments within V6.1 applications and attachments to service applications that are deployed to a V6.1 server. Additionally, these default bindings are used for V6.1 attachments unless they are overridden at the attachment point by an application specific binding or a V6.1 server default binding.

Results

When you complete these steps, the server default bindings are defined and all policy set attachments that specify use of the default binding for your web service applications that are deployed to the server will use server level default bindings.

Example

Suppose you have configured an application server, `server1`, and you have deployed several web service applications to the `server1` application server. Because these applications have similar security and quality of service requirements and you plan for them to share security configuration, you want to define the default bindings for policy set attachments to service providers and clients using the `server1`.

Suppose also that you want to modify the provided general provider binding, **Provider sample**. You can copy and modify this provided sample to take advantage of existing bindings.

1. Copy and modify the provided **Provider sample** and **Client sample** to meet your security and quality of service requirements. Include binding configuration for all policy types.

- Click **Services > Policy sets > General provider policy set bindings**. Select **Provider sample > copy**. Name the new general provider binding, MyServiceProviderbinding , and provide a description for the new binding.
- Click **Services > Policy sets > General client policy set bindings**. Select **Client sample > copy**. Name the new general client binding, MyServiceClientbinding, and provide a description for the new binding.

2. Locate *server1* in the Application servers collection and click the instance. From the administrative console, select **Servers > Server Types > WebSphere application servers** , and click the *server1* instance.
3. Click **Default policy set bindings**.
4. Select the bindings that you want to use for your provider and client policy set attachments. In this example, select your customized general bindings, MyServiceProviderbinding and MyServiceClientbinding.
5. Click **Apply** or **OK** to submit your changes.
6. Click **Save** to save your changes to the master configuration.

Each time you attach a policy set to a service or client deployed to the *server1* application server, it is initially set to use the specified bindings.

.

What to do next

After setting server default bindings, you can start deploying services to the server and start attaching policy sets. Alternatively, you might already have services deployed to the server, and the server is using the global default bindings because there is no server default binding. Now that you have set server default bindings, ensure that the server default bindings are used for the service messages as specified.

Server default binding settings:

Use this page to specify the server default bindings if you want to override the default bindings that are specified for the cell (global security) or the security domain to which the server is deployed.

To view this administrative console page, complete the following actions:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. From Security, click **Default policy set bindings**.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Policy set bindings for servers

To understand default policy set bindings, it is important to first understand policy set bindings.

Policy set bindings contain platform-specific information, such as keystore, authentication information or persistent information that is required by a policy set attachment. A policy set attachment is a policy set that is attached to an application resource. In WebSphere Application Server Version 7.0 and later, there are two types of bindings, general bindings and application specific bindings.

There are two types of general bindings, general service provider bindings and general service client bindings. You can configure one or more general service provider bindings and one or more general service client bindings across a range of policy sets. Additionally, you can re-use these general bindings across applications and for trust service attachments. To define and manage general bindings, in the administrative console click **Services** > **Policy sets** > **General provider policy set bindings** or **Services** > **Policy sets** > **General client policy set bindings**. The general service provider and client bindings

have independent settings that you can customize to meet the needs of your environment. To learn more about general bindings, read about defining and managing policy set bindings.

You can create application specific bindings when you attach a policy set to an application resource. These bindings are specific to and defined to the characteristics of the policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have limited reuse across policy sets. To assign application specific bindings to an application for service providers, in the administrative console click **Applications** > **Applications Types** > **WebSphere enterprise applications** > ***application_name*** > **Service provider policy sets and bindings** > **Assign Binding** > **New Application Specific Binding**. To assign application specific bindings to an application for service clients, in the administrative console click **Applications** > **Applications Types** > **WebSphere enterprise applications** > ***application_name*** > **Service client policy sets and bindings** > **Assign Binding** > **New Application Specific Binding**.

Default policy set bindings for servers

Note: In WebSphere Application Server Version 7.0 and later, the security model was enhanced to a domain-centric security model instead of a server-based security model. The configuration of the default global security (cell) level and default server level bindings has also changed in this version of the product. In the WebSphere Application Server Version 6.1 Feature Pack for Web Services, you can configure one set of default bindings for the cell and optionally configure one set of default bindings for each server. In Version 7.0 and later, you can configure one or more general service provider bindings and one or more general service client bindings. After you have configured general bindings, you can specify which of these bindings is the global default binding. You can also optionally specify general binding that are used as the default for an application server or a security domain.

General service provider and client bindings are not linked to a particular policy set, and they provide configuration information that you can reuse across multiple applications. You can create and manage general provider and client policy set bindings and then select one of each binding to use as the default for an application server. Setting the server default bindings is useful if you want the services that are deployed to a server to share binding configuration. You can also accomplish this sharing of binding configuration by assigning the binding to each application deployed to the server or by assigning a security domain with a default binding to your server.

You can specify default bindings for your service provider or client that are used at the cell (global security) level, for a security domain, or for a particular server. The default bindings are used in the absence of an overriding binding specified at a lower scope. The application server uses the following order of precedence, from lowest to highest, when determining which default bindings to use:

1. Server level default
2. Security domain level default
3. Cell (global security) default

The general bindings that are provided with the product are initially set as the cell (global security) default bindings. The default service provider binding and the default service client bindings are used when no application specific bindings or trust service bindings are assigned to a policy set attachment. If you do not want to use the provided **Provider sample** as the default service provider binding, you can select an existing general provider binding or create a new general provider binding to meet your business needs. Likewise, if you do not want to use the provided **Client sample** as the default service client binding, you can select an existing general client binding or create a new general client binding. To specify a cell (global security) default bindings, in the administrative console click **Services** > **Policy sets** > **Default policy set bindings**. For environments with multiple security domains, you can optionally choose the

general provider and general client bindings that you want to use as the default bindings for a domain. To learn more about default policy set bindings for the cell (global security), see the default policy set bindings documentation.

In addition to choosing default bindings for the cell (global security), you can also choose the general provider and general client bindings that you want to use as the default bindings for a server. Use this page to choose the default bindings for a server from the administrative console. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** and then from Security, click **Default policy set bindings**. If you do not choose a general binding as the default for a server, the default bindings for the domain in which the server resides is used. If you do not choose a binding as the default for a domain, the default bindings for the cell (global security) is used. You must choose a default service provider and default service client bindings for the cell. The general bindings that are included with the product are initially set as the cell (global security) default bindings. You cannot delete a binding that has been selected as the default binding for server, a domain, or the cell. Before you delete the binding, you must select a different binding as the default or choose to use the defaults for the cell (global security).

Note:

If you have an application that contains one or more application specific bindings that are configured at the WebSphere Application Server Version 6.1 level, this application is a V6.1 application. If you have applications that are deployed to V6.1 servers within the Version 7.0 or later application server environment, or you have V6.1 applications that are deployed to V7.0 or later versions of the application server, you can specify Version 6.1 default policy set bindings for the cell. These bindings are used for both client and provider policy set attachments within V6.1 applications and attachments to service applications that are deployed to a V6.1 server. Additionally, these default bindings are used for V6.1 attachments unless they are overridden at the attachment point by an application specific binding or a V6.1 server default binding. You can upgrade V6.1 bindings to the bindings that are used by WebSphere Application Server V7.0 and later versions. Use the upgradeBindings command using the wsadmin tool to upgrade the bindings level, if the V6.1 application is not installed on WebSphere Application Server V6.1.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Server default provider binding:

Specifies the default server binding. Select the name of the binding you want to use as the default for service providers deployed to this server.

If you are using multiple security domains, the name of the security domain to which the binding is scoped is specified with the name of the server default provider binding. Only bindings that are scoped to global security or to the security domain to which the server is assigned are available in the list.

Note: It is a best practice to specify a default binding that includes all of the policy types. This practice ensures that your default service provider binding has the necessary configuration for all policy types that you want to use.

Server default client binding:

Specifies the default client binding. Select the name of the binding you want to use as the default for service clients deployed to this server.

If you are using multiple security domains, the name of the security domain to which the binding is scoped is specified with the name of the server default client binding. Only bindings that are scoped to global security or to the security domain to which the server is assigned are available in the list.

Note: It is a best practice to specify a default binding that includes all of the policy types. This practice ensures that your default service client binding has the necessary configuration for all policy types that you want to use.

Version 6.1 default bindings:

If you have a Version 6.1 application that you want to use in the Version 7.0 and later application server environment and you want to specify Version 6.1 default bindings for this server, you can click this link to specify Version 6.1 default bindings.

Server Version 6.1 default policy set bindings:

Use this page to specify the server Version 6.1 default policy set bindings for this server. These default bindings are used for client and provider policy set attachments for applications that are deployed to Version 6.1 servers, and for Version 6.1 applications that are deployed to Version 7.0 servers. The default bindings are used for Version 6.1 attachments unless overridden at the attachment point.

To view this administrative console page, complete the following actions:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***
2. Under Security, click **Default policy set bindings**
3. Click **Version 6.1 default policy set bindings**

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Version 6.1 default policy set bindings

You can install a WebSphere Application Server Version 6.1 Feature Pack for Web Services application within the WebSphere Application Server Version 7.0 and later environment. If your application contains Version 6.1 custom bindings, the application is defined as a Version 6.1 application to the application server. Additionally, if your application is installed on a WebSphere Application Server Version 6.1 Feature Pack for Web Services server within the WebSphere Application Server Version 7.0 and later environment, the custom binding is created as a Version 6.1 custom binding.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Bindings configuration name:

Specifies the name of the policy set bindings configuration. This field is read-only for default policy set bindings.

Button	Resulting action
Add	Adds a policy to the collection.
Delete	Removes a policy from the collection.

Setting default policy set bindings

You can set provider and client default policy set bindings that are used as the global security default policy set bindings. The specified global security default bindings apply to all web services unless the bindings are overridden at the attachment point, at the server, or at a security domain.

Before you begin

You must first install and configure an application server. After the application server is installed, you must install a Java API for XML-Based Web Services (JAX-WS) application onto your server. Now, you are

ready to attach a policy set to the web service application. You can define and manage general service provider and client policy set bindings for your web service resource by using the administrative console.

About this task

Policy set bindings for servers

After you understand policy set bindings, then it is easier to understand how the default bindings are used.

Policy set bindings contain platform-specific information, such as keystore, authentication information or persistent information that is required by a policy set attachment. A policy set attachment is a policy set that is attached to an application resource. Starting with WebSphere Application Server Version 7.0 and later, there are two types of bindings, general bindings and application specific bindings.

There are two types of general bindings, *general service provider bindings* and *general service client bindings*. You can configure one or more general service provider bindings and one or more general service client bindings and then use them across a range of policy sets. Additionally, you can re-use these general bindings across applications and for trust service attachments. To define and manage general bindings, in the administrative console click **Services > Policy sets > General provider policy set bindings** or **Services > Policy sets > General client policy set bindings**. The general service provider and client bindings have independent settings that you can customize to meet the needs of your environment.

You can create *application specific bindings* when you attach a policy set to a web service application resource. These bindings are specific to and defined by to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have limited reuse across policy sets. To assign application specific bindings to an application for service providers, in the administrative console click **Applications > Applications Types > WebSphere enterprise applications > *application_name* > Service provider policy sets and bindings**. Select a web service resource with an attached policy and click **Assign Binding > New Application Specific Binding**. To assign application specific bindings to an application for service clients, in the administrative console click **Applications > Applications Types > WebSphere enterprise applications > *application_name* > Service client policy sets and bindings** . Select a web services resource with an attached policy and click **Assign Binding > New Application Specific Binding**. You can additionally assign application specific bindings for service providers or service clients using the administrative console and click **Services > Service providers > *application_name*** or **Services > Service clients > *application_name*** and then select a web services resource with an attached policy and assign your bindings.

To learn more about general bindings or application specific bindings, read about defining and managing policy set bindings.

Default policy set bindings

Note: In WebSphere Application Server Version 7.0 and later, the security model was enhanced to a domain-centric security model instead of a server-based security model. The configuration of the default global security (cell) level and default server level bindings has also changed in this version of the product. In the WebSphere Application Server Version 6.1 Feature Pack for Web Services, you can configure one set of default bindings for the cell and optionally configure one set of default bindings for each server. In Version 7.0 and later, you can configure one or more general service provider bindings and one or more general service client bindings. After you have configured general bindings, you can specify which of these bindings is the global default binding. You can also optionally specify general binding that are used as the default for an application server or a security domain.

General service provider and client bindings are not linked to a particular policy set, and they provide configuration information that you can reuse across multiple applications. You can create and manage general provider and client policy set bindings and then select one of each binding type to use as the default for an application server. Setting the server default bindings is useful if you want the services that are deployed to a server to share binding configuration. You can also share binding configuration by either assigning the binding to each application that is deployed to the server or by setting default bindings for a security domain and assigning the security domain to one or more servers.

You can specify default bindings for your service provider or client that are used at the global security (cell) level, for a security domain, for a particular server. The default bindings are used in the absence of an overriding binding specified at a lower scope. The following list is the order of precedence from lowest to highest that the application server uses to determine which default bindings to use:

1. Server level default
2. Security domain level default
3. Global security (cell) default

The sample general bindings that are provided with the product are initially set as the global security (cell) default bindings. The default service provider binding and the default service client bindings are used when no application specific bindings or trust service bindings are assigned to a policy set attachment. For trust service attachments, the default bindings are used when no trust specific bindings are assigned. If you do not want to use the provided **Provider sample** as the default service provider binding, you can select an existing general provider binding or create a new general provider binding to meet your business needs. Likewise, if you do not want to use the provided **Client sample** as the default service client binding, you can select an existing general client binding or create a new general client binding. To specify your global security (cell) default bindings, in the administrative console click **Services > Policy sets > Default policy set bindings**. For environments with multiple security domains, you can optionally choose the general provider and general client bindings that you want to use as the default bindings for a domain.

In addition to choosing default bindings for the global security (cell), you can also choose the general provider and general client bindings that you want to use as the default bindings for a server. This is only necessary if you want to use different default bindings for a particular server than those used by the other servers in the security domain or cell. To choose the default bindings for a server from the administrative console, click **Servers > Server Types > WebSphere application servers >** *server_name* and then from Security, click **Default policy set bindings**. If you do not choose a general binding as the default for a server, the default bindings for the domain in which the server resides is used. If you do not choose a binding as the default for a domain, the default bindings for the global security (cell) are used. You must choose a default service provider and default service client bindings for the cell. The general bindings that are included with the product are initially set as the global security (cell) default bindings. You cannot delete a binding that is used as part of any policy set attachment or specified as the default binding for the server, a domain, or the cell. To learn more about defining default bindings for a server, see the server default bindings documentation.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. Open the administrative console.
2. To set global security default policy set bindings, select **Services > Policy sets > Default policy set bindings**.
3. Select the default service provider binding. The default service provider binding is used as the default for policy set attachments unless the provider or client binding is overridden at the attachment point, at the server, or at a security domain. The default setting is **Provider sample**.

4. Select the default service client binding. If you specify a default service client binding, the selected binding overrides the default bindings that are specified for the cell or the security domain to which the server is deployed. The default setting is **Client sample**.
5. If multiple security domains are enabled, you can view the default provider bindings and the default client bindings for each security domain that is defined in the security domain default bindings collection. You can select the security domain name link if you want to access the domain and select different default bindings. Additionally, you can select the default provider binding links or the default client binding links to access the default bindings and select different default binding settings.
6. Click **Apply** to apply selected bindings as the global default bindings.
7. Click **Save** to save your changes to the master configuration.
8. (optional) If you are using a Version 6.1 application, you can specify server V6.1 default policy set bindings. To set these bindings, select **Services > Policy sets > Default policy set bindings > Version 6.1 default policy set bindings**.

   Note:

   If you have an application that contains one or more application specific bindings that are configured at the WebSphere Application Server Version 6.1 level, this application is a V6.1 application. If you have applications that are deployed to V6.1 servers within the Version 7.0 or later application server environment, or you have V6.1 applications that are deployed to V7.0 or later versions of the application server, you can specify Version 6.1 default policy set bindings for the cell. These bindings are used for both client and provider policy set attachments within V6.1 applications and attachments to service applications that are deployed to a V6.1 server. Additionally, these default bindings are used for V6.1 attachments unless they are overridden at the attachment point by an application specific binding or a V6.1 server default binding. You can upgrade V6.1 bindings to the bindings that are used by WebSphere Application Server V7.0 and later versions. Use the upgradeBindings command using the wsadmin tool to upgrade the bindings level, if the V6.1 application is not installed on WebSphere Application Server V6.1.

Results

When you complete these steps, you have defined your global security (cell) default policy set bindings and domain default policy set bindings, if applicable.

Example

Suppose that you do not want to use the provided general provider binding, **Provider sample**, as your default service provider binding. To take advantage of existing bindings, you can copy and modify the **Provider sample** to meet your business needs. This example assumes that your server environment has SecurityDomain1 and SecurityDomain2 defined.

1. Copy and modify the provided **Provider sample** general service provider binding. Click **Services > Policy sets > General provider policy set bindings**. Select **Provider sample > copy**. Name the new general provider binding, MyServiceProviderbinding, and provide a description for the new binding.
2. Copy and modify the provided **Client sample** general service client binding. Click **Services > Policy sets > General client policy set bindings**. Select **Client sample > copy**. Name the new general client binding, MyServiceClientbinding, and provide a description for the new binding.
3. To specify the default policy set bindings for your global security (cell) and for your domains, click **Services > Policy sets > Default policy set bindings**. From this page, select MyServiceProviderbinding as the default service provider binding, and select MyClientProviderbinding as the default service client binding.
4. Click **Apply** and **Save** to save your changes to the master configuration.

Assigning a domain default binding is optional. Generally, you assign domain default policy set bindings only when you want the servers in the domain to use different default bindings than the rest of the cell. In this example, suppose you have defined another general provider binding, MyServiceProviderbinding2, and you want to specify this binding as the domain default binding for your SecurityDomain1 domain.

1. From the security domain default bindings collection click **SecurityDomain1 > Default policy set bindings**. From this page, you can select MyServiceProviderbinding2 as the service provider domain default binding.
2. Click **Apply** and **Save** to save your changes to the master configuration.

Default policy set bindings collection:

Use this page to specify the service provider and client default bindings. The specified service provider and client bindings are used at the cell (global security) level unless these specified bindings are overridden at the attachment point, at the server, or at a security domain.

To view this administrative console page, click **Services** > **Policy sets** > **Default policy set bindings**.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Policy set bindings for servers

To understand default policy set bindings, it is important to first understand policy set bindings.

Policy set bindings contain platform-specific information, such as keystore, authentication information or persistent information that is required by a policy set attachment. A policy set attachment is a policy set that is attached to an application resource. There are two types of bindings, general bindings and application specific bindings.

There are two types of general bindings, general service provider bindings and general service client bindings. You can configure one or more general service provider bindings and one or more general service client bindings across a range of policy sets. Additionally, you can re-use these general bindings across applications and for trust service attachments. To define and manage general bindings, in the administrative console click **Services > Policy sets > General provider policy set bindings** or **Services** > **Policy sets** > **General client policy set bindings**. The general service provider and client bindings have independent settings that you can customize to meet the needs of your environment. To learn more about general bindings, read about defining and managing policy set bindings.

You can create application specific bindings when you attach a policy set to an application resource. These bindings are specific to and defined to the characteristics of the policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application specific bindings have limited reuse across policy sets. To assign application specific bindings to an application for service providers, in the administrative console click **Applications** > **Applications Types** > **WebSphere enterprise applications** > ***application_name*** > **Service provider policy sets and bindings** > **Assign Binding** > **New Application Specific Binding**. To assign application specific bindings to an application for service clients, in the administrative console click **Applications** > **Applications Types** > **WebSphere enterprise applications** > ***application_name*** > **Service client policy sets and bindings** > **Assign Binding** > **New Application Specific Binding**.

Default policy set bindings for servers

Note: In WebSphere Application Server Version 7.0 and later, the security model was enhanced to a domain-centric security model instead of a server-based security model. The configuration of the default global security (cell) level and default server level bindings has also changed in this version of the product. In the WebSphere Application Server Version 6.1 Feature Pack for Web Services, you can configure one set of default bindings for the cell and optionally configure one set of default

bindings for each server. In Version 7.0 and later, you can configure one or more general service provider bindings and one or more general service client bindings. After you have configured general bindings, you can specify which of these bindings is the global default binding. You can also optionally specify general binding that are used as the default for an application server or a security domain.

General service provider and client bindings are not linked to a particular policy set, and they provide configuration information that you can reuse across multiple applications. You can create and manage general provider and client policy set bindings and then select one of each binding to use as the default for an application server. Setting the server default bindings is useful if you want the services that are deployed to a server to share binding configuration. You can also accomplish this sharing of binding configuration by assigning the binding to each application deployed to the server or by assigning a security domain with a default binding to your server.

You can specify default bindings for your service provider or client that are used at the cell (global security) level, for a security domain, or for a particular server. The default bindings are used in the absence of an overriding binding specified at a lower scope. The application server uses the following order of precedence, from lowest to highest, when determining which default bindings to use:

1. Server level default
2. Security domain level default
3. Cell (global security) default

The general bindings that are provided with the product are initially set as the cell (global security) default bindings. The default service provider binding and the default service client bindings are used when no application specific bindings or trust service bindings are assigned to a policy set attachment. If you do not want to use the provided **Provider sample** as the default service provider binding, you can select an existing general provider binding or create a new general provider binding to meet your business needs. Likewise, if you do not want to use the provided **Client sample** as the default service client binding, you can select an existing general client binding or create a new general client binding. To specify a cell (global security) default bindings, in the administrative console click **Services** > **Policy sets** > **Default policy set bindings**. For environments with multiple security domains, you can optionally choose the general provider and general client bindings that you want to use as the default bindings for a domain. To learn more about default policy set bindings for the cell (global security), see the default policy set bindings documentation.

In addition to choosing default bindings for the cell (global security), you can also choose the general provider and general client bindings that you want to use as the default bindings for a server. Use this page to choose the default bindings for a server from the administrative console. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** and then from Security, click **Default policy set bindings**. If you do not choose a general binding as the default for a server, the default bindings for the domain in which the server resides is used. If you do not choose a binding as the default for a domain, the default bindings for the cell (global security) is used. You must choose a default service provider and default service client bindings for the cell. The general bindings that are included with the product are initially set as the cell (global security) default bindings. You cannot delete a binding that has been selected as the default binding for server, a domain, or the cell. Before you delete the binding, you must select a different binding as the default or choose to use the defaults for the cell (global security).

Note:

If you have an application that contains one or more application specific bindings that are configured at the WebSphere Application Server Version 6.1 level, this application is a V6.1 application. If you have applications that are deployed to V6.1 servers within the Version 7.0 or later application server environment, or you have V6.1 applications that are deployed to V7.0 or later versions of the application server, you can specify Version 6.1 default policy set bindings for the cell. These bindings are used for both client and provider policy set attachments within V6.1

applications and attachments to service applications that are deployed to a V6.1 server. Additionally, these default bindings are used for V6.1 attachments unless they are overridden at the attachment point by an application specific binding or a V6.1 server default binding. You can upgrade V6.1 bindings to the bindings that are used by WebSphere Application Server V7.0 and later versions. Use the upgradeBindings command using the wsadmin tool to upgrade the bindings level, if the V6.1 application is not installed on WebSphere Application Server V6.1.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Default service provider binding:

Specifies the default service provider binding that is used as the default for policy set attachments. You can override this default at the attachment point or by a lower level default.

Note: It is a best practice to specify a default binding that includes all of the policy types. This practice ensures that your default service provider binding has the necessary configuration for all policy types that you want to use..

Default service client binding:

Specifies the default service client binding that is used as the default for policy set attachments. You can override this default at the attachment point or by a lower level default.

Note: It is a best practice to specify a default binding that includes all of the policy types. This practice ensures that your default service client binding has the necessary configuration for all policy types that you want to use.

Security domain default bindings:

Specifies a collection of security domain default bindings. This collection only displays if you are using multiple security domains.

If you are using multiple security domains, this collection displays the default client and provider bindings for each security domain. You can select the security domain name link if you want to access the domain and select different default bindings.

Security domain:

Specifies the name of a security domain.

Default provider binding:

Specifies the default service provider binding for the security domain. You can view the default provider binding settings by clicking on the name of default provider binding.

Default client Binding:

Specifies the default client binding for the security domain. You can view the default client binding settings by clicking on the name of default client binding.

Version 6.1 default bindings:

If you have a Version 6.1 application that you want to use in the Version 7.0 application server environment and you want to specify Version 6.1 default bindings for this server, you can click this link to specify Version 6.1 default bindings.

Version 6.1 default policy set bindings:

Use this page to specify Version 6.1 default policy set bindings for the cell (global security). These bindings are used for both client and provider policy set attachments within Version 6.1 applications and attachments to service applications that are deployed to a Version 6.1 server. These default bindings are used for Version 6.1 attachments unless they are overridden at the attachment point or by a Version 6.1 server default binding.

To view this administrative console page when you are specifying Version 6.1 cell level default policy set bindings, complete the following actions:

1. Click **Services** > **Policy sets** > **Default policy set bindings**
2. Click **Version 6.1 default policy set bindings**

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Version 6.1 default policy set bindings

You can install a WebSphere Application Server Version 6.1 Feature Pack for Web Services application within the WebSphere Application Server Version 7.0 and later environments. If your application contains Version 6.1 custom bindings, the application is defined as a V6.1 application to the application server. Additionally, if your application is installed on a WebSphere Application Server Version 6.1 Feature Pack for Web Services server within the WebSphere Application Server Version 7.0 and later environments, the custom binding is created as a V6.1 custom binding.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Bindings configuration name:

Specifies the name of the policy set bindings configuration. This field is read-only for default policy set bindings.

Policies – HTTP transport:

Links to the HTTP transport policy settings page where you define the HTTP transport settings. The HTTP features and HTTP connection polices are applied to outbound messages. The response listener policy is enforced on inbound messages.

Policies – SSL transport:

Links to the SSL transport policy configuration settings page where you define the SSL transport settings.

Policies – WS-Addressing:

Links to the configuration settings page for the WS-Addressing policy. In a network deployment environment, use this page to enable or disable workload management. Otherwise, you can attach the WS-Addressing policy set to service resources. No additional configuration is required.

Policies – WS-ReliableMessaging:

Links to the page where you configure the WS-ReliableMessaging bindings.

Policies – WS-Security:

Links to the WS-Security policy set bindings settings page where the WS-Security binding are configured.

Domain default bindings settings:

Use this page to specify the default policy set bindings for this security domain.

To view this administrative console page, complete the following actions:
1. Click **Security domains >** *domain_name*
2. From Web Services Bindings, click **Default policy set bindings**

Alternatively, you can view this administrative console page when multiple security domains are defined:
1. Click **Services > Policy sets > Default policy set bindings>*domain_name***
2. In the Security Domain Default Bindings collection, click *domain_name*
3. From Web Services Bindings, click **Default policy set bindings**

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Domain default provider binding:

Specifies the default provider binding for a specific security domain. You can specify `None` if you want to use the server or global default bindings for services deployed to this server. If you specify a server default binding, this binding takes precedence over the default bindings that are specified for the security domain to which the server is deployed. The name of the security domain to which each binding is scoped is displayed to the right of the name of the binding.

Note: It is a best practice to specify a binding that includes all of the policy types. This practice ensures that your default service provider binding has the necessary configuration for all policy types that you want to use.

Domain default client binding:

Specifies the default client binding for a specific security domain. You can specify `None` if you want to use the server or global default bindings for services deployed to this server. If you specify a server default binding, this binding will take precedence over the default bindings specified for the security domain to which the server is deployed. The name of the security domain to which each binding is scoped is displayed to the right of the name of the binding.

Note: It is a best practice to specify a binding that includes all of the policy types. This practice will ensure that your default client binding has the necessary configuration for all policy types that you want to use.

Modifying default bindings at the server level for policy sets

You can define default bindings for HTTP transport, JMS transport, Secure Sockets Layer (SSL) transport, WS-Addressing, WS-ReliableMessaging, and WS-Security policies.

Before you begin

To create or modify server default bindings, you must first install and configure an application server.

About this task

You can modify default cell bindings that are delivered with the product. Default bindings are not linked to a particular policy set and they provide default settings that might be used for sharing configuration information across multiple applications.

You can create default bindings for individual application servers and customize those bindings to meet your requirements. The policy set bindings are optional. If a policy set binding is not defined at the server level, then the default cell level bindings are used. To modify these default bindings, complete the following steps:

Procedure

1. Open the administrative console.
2. To see which bindings are defined as the defaults for the cell or server, click **Services > Policy sets > Default policy sets bindings**.

   After seeing which bindings are defaults, you can select and edit them from the General provider or General client policy set bindings pages.
3. To edit the default bindings, see “Setting default policy set bindings” on page 2697.

Results

When you finish these steps, the default policy set bindings are modified. If you created default bindings at the server level, those default bindings are used instead of the default bindings at the cell level.

What to do next

You can create application specific bindings for your policy sets. Read about creating application specific bindings for policy set attachments.

Reassigning bindings to policy sets attachments

After you create a custom attachment binding, you can reassign that binding to another service artifact if necessary. You can reset a service artifact, such as an application, service, or endpoint to use the inherited bindings or default bindings.

Before you begin

To reassign an application specific binding or a default binding, there must be an assigned binding for the service artifact.

About this task

You can assign bindings to policy sets and other service artifacts. If you created a policy set and attached it to a service artifact, you can reassign the binding that was initially assigned to that policy set.

Procedure

1. Open the administrative console.
2. To reassign a binding for your service provider, click **Applications > Application Types > WebSphere enterprise applications > *application_name* > Service provider policy sets and bindings**.

   To reassign a binding for your service client, click **Applications > Application Types > WebSphere enterprise applications > *application_name* > Service client policy sets and bindings**.
3. Select the check box for the service artifact with an attachment binding.
4. Select **Assign Binding**.
5. Select the binding that you want to assign.

6. Click **Save**, to save your changes to the master configuration.

Results

The service artifact now inherits the policy set bindings that are indicated in either the **Service provider policy sets and bindings** or **Service client policy sets and bindings** panel.

Restriction: Reassigning the binding for a service artifact only reassigns it for that service artifact and any inherited artifacts. It does not reassign the binding for other non-related and non-inherited attachments to the same policy set or to a different policy set.

Example

If you have the application, `app1` with an attached policy set `Username WSSecurity default`, and you want to define custom bindings, then perform the following steps:

1. Locate the application in the **Applications > Application Types > WebSphere enterprise applications >** ***application_name*** collection.
2. Click the **app1** application.
3. Click the **Service provider policy sets and bindings** link or the **Service client policy sets and bindings** link.
4. Select the check box for the**app1** application where the custom attachment binding is defined.
5. Click **Assign Binding** and select **New**.
6. Provide a name for the application specific bindings.
7. Add required policies to the new application specific bindings.
8. Configure the bindings.
9. Click **Save**, to save your changes to the master configuration.

What to do next

Restart the application that contains the policy set where you reassigned the bindings.

Policy set bindings settings

Use this page to view or define general, application specific, or trust service specific bindings configuration information for policies that you can associate with the selected policy set. This bindings configuration information is specific to a system. Use the links on this page to work with bindings for each specific policy.

To view this administrative console page when you are creating or editing a trust service specific binding, click **Services** > **Trust service** > **Trust service attachments**. Select a binding by clicking the binding name in the table if the binding has been assigned to an attachment. To create a new trust service specific binding, click **Assign binding** > **New Trust Service Specific Binding**.

To view this administrative console page when you are editing a general binding, click **Services** > **Policy sets** > **General provider policy set bindings** or **Services** > **Policy sets** > **General client policy set bindings**.

To view this administrative console page when you are creating or editing an application specific binding, complete the following actions:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications**.
2. Select an application that contains web services. The application must contain a service provider or a service client.
3. Click **Service provider policy sets and bindings** or **Service client policy sets and bindings** in the Web Services Properties section.

4. Select a binding by clicking the binding name in the table if the binding has been assigned to an attachment. You must have previously attached a policy set and assigned an application specific binding.
5. [Optional] To edit a default cell binding or default server binding, click either**Services** > **Policy sets** > **General provider policy set bindings** or **Services** > **Policy sets** > **General client policy set bindings**.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

About Policy set bindings

Policy set bindings contain platform-specific information, like keystore, authentication information or persistent information, required by a policy set attachment. Each policy set attachment to a service provider or service client must have exactly one binding. When you create a policy set attachment, the general default bindings are used initially. When general bindings are used in association with a policy set attachment, the cell-level general bindings are applied at run time. If application server level bindings exist, the server-level general bindings override the cell-level definition. General bindings specify configuration for both service client and service provider attachments and the general bindings are not tailored to a specific policy set or application. When you define server-level general bindings, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP Transport, that you want to override the general binding and you must fully configure the bindings for each policy that you have added.

An application specific binding is a named binding that you create. Application specific bindings enable you to provide platform-specific configuration information for specific policy set attachments. When you create an application specific binding, the available binding configuration options are tailored to the definitions in the attached policy set. You can reuse application specific bindings for multiple service resources within an application. For example, if you create a trust service specific binding, that binding can be reused only for trust service attachments. When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP Transport, that you want to override the general binding and you must fully configure the bindings for each policy that you have added.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Bindings configuration name:

Specifies the name of the policy set bindings configuration. The binding name is not editable when you are editing a binding. When you are creating a new binding, you must specify the binding name.

Note: If you are running a Version 6.1 application, the Binding configuration name is displayed as Version 6.1 default policy set bindings.

Use the following actions to create, edit, or delete policy set bindings.

Button	Resulting action
Add	Adds the selected policy set binding to the application.
Delete	Removes the selected policy set binding from the application.

Policies – HTTP transport:

Links to the HTTP transport policy configuration settings page where you define the HTTP transport settings. The HTTP features and HTTP connection polices are applied to outbound messages. The response listener policy is enforced on inbound messages.

Policies – SSL transport:

Links to the SSL transport policy configuration settings page where you define the SSL transport settings.

Policies – JMS transport:

Links to the JMS transport policy configuration settings page where you define the JMS transport settings.

Policies – WS-Addressing:

Links to the configuration settings page for the WS-Addressing policy. In a WebSphere Application Server, Network Deployment environment, use this page enable or disable workload management. Otherwise, you can attach the WS-Addressing policy set to service resources. No additional configuration is required.

Policies – WS-ReliableMessaging:

Links to the panel where the WS-ReliableMessaging bindings are configured.

Policies – WS-Security:

Links to the WS-Security policy set bindings settings page where the WS-Security bindings are configured.

Policies – Custom properties:

Links to the Custom properties policy set bindings settings page where the Custom properties bindings are configured. The CustomProperties binding is only supported for service clients.

Web Services Addressing policy set binding

Use this page to modify the endpoint reference binding for Web Services Addressing (WS-Addressing). The product enforces this binding on JAX-WS web service applications that use WS-Addressing. This panel applies only to the WebSphere Application Server, Network Deployment version of the product.

To view this administrative console page, click **Services** > **Policy Sets** > **Default policy set bindings** > **WS-Addressing**.

Prevent workload management of referenced endpoints in clusters:

By default, when an application uses the WS-Addressing API createEndpointReference(service, port) method to create an endpoint reference in a cluster environment, that endpoint reference is workload managed. Select this option to prevent the workload management of such endpoint references.

If you select this option, messages that are targeted at such an endpoint reference are always sent to the node on which the endpoint reference was created. This behavior is useful when the endpoint contains in-memory state that is held on a single node, for example, in Web Services Resource Framework (WS-RF) applications.

Note: Although the endpoint reference always represents the same endpoint, the state that is held at that endpoint is not reliable. If the node that is hosting the endpoint fails, the state might be lost. Note also that if the node fails, the endpoint reference is no longer valid.

Data type	Check box
Default	Cleared

Range

Cleared
Endpoint references that are created by the application in a cluster environment are workload managed.

Selected
Endpoint references that are created by the application in a cluster environment are not workload managed.

Attaching a policy set to a service artifact

Attach a policy set to a service artifact, such as an application, service, endpoint or operation, to define the quality of services that are supported. Policy sets can define the policies for WS-Addressing, WS-Security, WS-ReliableMessaging, WS-Transaction, HTTP transport, Java Messaging Service (JMS) transport, and Secure Sockets Layer (SSL) transport.

Before you begin

Before you can start this task, you must deploy an application containing web services. Also, if none of the default policy sets contain the necessary policy definitions, then you must create a custom policy set with the necessary definitions.

About this task

Develop a web service that contains each of the necessary artifacts, and deploy your web services application into your application server instance. Now you can attach policy sets to your service artifacts, such as an application, service, or endpoint.

To attach a policy set to a service artifact, perform the following steps:

Procedure

1. Open the administrative console.
2. To attach a policy set to a service provider, click **Applications > Enterprise applications** > ***application_name*** > **Service provider policy sets and bindings**.

   To attach a policy set to a service client, click **Applications > Enterprise applications** > ***application_name*** > **Service client policy sets and bindings**.
3. Select the check box for the service artifact.
4. Select **Attach** to display a list of available policy sets to attach. Select a policy set from the list.
5. Click **Save**, to save your changes to the master configuration.
6. [Optional] To see what attachments are defined for a given policy set, select **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **Attached applications**.

Results

When you finish these steps, a policy set is attached to the service artifact.

Example

If you have the application, `app1` and you want to attach the policy set, `WSSecurity default`, then perform the following steps:

1. Locate the `app1` application in the **Applications** > **Enterprise applications** collection.
2. Click the **app1** application.

3. Click the **Service provider policy sets and bindings** link or the **Service client policy sets and bindings** link.
4. Select the check box for the service artifact where the policy set is to be attached.
5. Click **Attach**. Select **WSSecurity default** policy set.
6. Click **Save**, to save your changes to the master configuration.

What to do next

You can create custom bindings for policy set attachments. Read about creating custom bindings for policy set attachments.

You can configure the service client or service provider to share their policies. Read about using WS-Policy to exchange policies in a standard format.

Managing policies in a policy set using the administrative console

When working with policy sets in the administrative console, you can customize the included policies to ensure message security. You can enable, disable, customize, add, or delete policies from a policy set. With your policy sets, you can define policies for WS-Addressing, WS-Security, WS-ReliableMessaging, WS-Transaction, HTTP transport, Java Messaging Service (JMS) transport, and Secure Sockets Layer (SSL) transport. The policies for all but WS-Security are relatively straightforward to define.

Before you begin

You can customize the policies for custom policy sets. The provided default policy sets cannot be edited. Make sure that you have created a copy of the default policy set or created a completely new policy set to specify the policies for this policy set.

About this task

Customize policies associated with a policy set using the administrative console.

Procedure

1. Click one of the following:
   - **Services > Policy sets > Application policy sets** >*policy_set_name*
   - **Services > Policy sets > System policy sets** >*policy_set_name*

   You can also click **Services > Policy sets > Application policy sets** > **New** or **Services > Policy sets > System policy sets** > **New**. Follow this path when you want to create a new policy set and the associated policy or policies.
2. Add a policy to a custom policy set. To do this, see “Adding policies to policy sets using the administrative console” on page 2712.
3. Enable a policy for a custom policy set. To do this, see “Enabling policies for policy sets using the administrative console” on page 2757.
4. Optional: You can also disable a policy from a custom policy set. To do this, see “Disabling policies from policy sets using the administrative console” on page 2757.
5. Optional: You can also delete a policy from a custom policy set. To do this, see “Deleting policies from policy sets using the administrative console” on page 2712.

Results

After you have customized your policies, the associated policy set can protect messages according to the policy or policies defined.

What to do next

If the policy set you have modified is an attached policy set, restart all affected applications to pick up the changes you made. If the policy set is unattached, then no further action is required.

Adding policies to policy sets using the administrative console

You can use the administrative console to add policies to policy sets. Adding and configuring policies to a policy set further defines the rules governing the policy set.

Before you begin

Before adding a policy to a policy set, check that it is not listed as disabled in the **Policies** table. A disabled policy can be listed for the policy set and only need to be enabled to be included in the policy set. If this is the case, the policy does not need to be added and it is not available to be added.

About this task

All policies are initially set to their default values. These values can be edited and the attributes changed using the administrative console.

Procedure

1. Click **Services > Policy sets > Application policy sets** > ***policy_set_name*** or **Services > Policy sets > System policy sets > *policy_set_name***. This displays a listing of available policies in the **Policies** table for the policy set selected. If you created a new policy set and did not copy and rename an existing policy set, this table might contain no policies and you must add them.
2. Add a policy. To add a policy not listed in the **Policies** table of the Policy set settings page, click the **Add** drop down button. This button displays a list of available policies that are not already listed in the **Policies** table and therefore not included in the policy set.
3. Click the policy to be added. The selected policy is added to the **Policies** table. Each policy you add has default settings that can be modified.

Results

After you have added a policy, it is then included in the policy set.

Example

If you were working on policy set ABC_ps (that you created and it therefore has no policies associated with it) you would click the ABC_ps name in the **Policy Sets** collection and then click the **Add** button in the **Policies** table on the **Policy set** settings window. The drop down list displays all available policies for you to choose from. You might decide you want to include the WS-Addressing and WS-Security policies for this policy set. You would select each of those from the list. Then, click the policy name in the table to edit the properties of these policies.

What to do next

If the modified policy set is an attached policy set, restart all affected applications to apply the changes. If the policy set is unattached, then no further action is required.

Deleting policies from policy sets using the administrative console

You can use the administrative console to delete policies from policy sets. Deleting the policy removes the policy that further defined the rules governing the policy set.

Before you begin

Before deleting a policy from a policy set, be sure that you no longer want the policy to be available to the policy set. Otherwise, consider disabling it so that it is still available to the policy set but not currently associated with the policy set. You can disable the policy and leave it available for the policy set by clicking the **Select** button by the policy to be disabled and then clicking the **Disable** button. That policy (and any configuration changes you have made to it) is still available to the policy set but is not currently included in the policy set.

About this task

To delete a policy from a policy set, use the administrative console.

Procedure

1. Click **Services > Policy sets > Application policy sets** > ***policy_set_name*** or **Services > Policy sets > System policy sets > *policy_set_name***. The Policy set settings page displays a listing of available policies in the **Policies** table of the Policy set settings page for the policy set selected. If you created a new policy set and did not copy and rename an existing policy set, this table might contain no policies to delete.
2. Delete the policy from the policy set.
   a. Click the **Select** box beside the policy that you want to delete in the **Policies** table of the Policy set settings page.
   b. Click the **Delete** button.
3. Verify that you want to delete the selected policy from the policy set.

Results

You have deleted the policy from the policy set.

Example

If you have a policy set containing a policy that you no longer want included in that policy set, you have two options: disable it or delete it. In some cases, you might want to disable the policy if you have configured attributes for it that you might want to enable at a later time. You can disable the policy and leave it available for the policy set by clicking the **Select** button by the policy to be disabled and then clicking the **Disable** button. But for this example, you have decided that the policy WS-Addressing is no longer valid for your custom policy set ABC_ps and you and you want to delete it. To delete it, click the **Select** button beside the WS-Addressing policy and then click **Delete**. Confirm that you want to delete the policy.

What to do next

You can now add other policies to the policy set or enable or disable existing policies.

Modifying policies using the administrative console

With your policy sets, you can define policies for WS-Addressing, WS-Security, WS-ReliableMessaging, WS-Transaction, HTTP transport, Java Messaging Service (JMS) transport, and Secure Sockets Layer (SSL) transport. The policies for all but WS-Security are relatively straightforward to define.

Before you begin

The provided default policy sets cannot be edited. You can create a copy of the default policy set or create a completely new policy set in order to specify the policies for it.

About this task

Polices are associated with policy sets, so you can change an instance of a policy for a particular policy set.

Procedure

1. Select the policy set containing the policy you want to modify. To customize the policies associated with a policy set, from the administrative console, click:
   - **Services > Policy sets > Application policy sets** >*policy_set_name*. Click a policy name in the Policies table.
   - Or **Services > Policy sets > System policy sets** >*policy_set_name*. Click a policy name in the Policies table.
2. Modify the settings. Depending on which Policy you choose, different settings can be modified.
3. [Optional] If the policy to be modified is not already in the Policies table, click **Add** and select a policy from the list to modify.
4. Save the changes you have made. Once you change the settings on a policy, you need to save the changes to return to the policy set.

Results

The policy set configuration is saved with the selected modifications.

Example

You have created a copy of the WSReliableMessaging persistent policy set and named it `WSRM_p1`. You want to change the settings on the WSReliableMessaging policy that is included, by default, in the copy of this policy set. So you click your `WSRM_p1` policy set from the Application policy sets window and then click the WSReliableMessaging policy from the Policies table. You can then alter the following settings:

Standard
The default setting is WSReliableMessaging 1.1.

Deliver messages in the order that they were sent
The default setting is false. Valid values are true or false.

Quality of service
The default is Unmanaged non-persistent

Save your changes and return to the Application policy sets window for the WSRM_p1 policy set.

What to do next

You can use the policy set as you have configured or you can change the bindings, attach or detach the policy set from applications.

Note: Because this policy set specifies managed persistent quality of service, you have to define bindings to the service integration bus and messaging engine that you want to use to manage the WS-ReliableMessaging state. You can attach and bind a WS-ReliableMessaging policy set to a web service application by using the administrative console or the wsadmin tool.

Configuring the WS-ReliableMessaging policy:

When working with policy sets in the administrative console, you can customize some policies.

Before you begin

This task assumes that you are working with a policy set to which the WS-ReliableMessaging policy has been added.

Do not edit the policies associated with the provided default policy sets. If you have to modify the reliable messaging policy settings, use a copy of a default policy set or create a new policy set.

At any stage - that is, before or after you have built your reliable web service application, or configured your policy sets - you can set a property that configures endpoints to only support clients that use reliable messaging. This setting is reflected by WS-Policy if engaged.

About this task

To configure the WS-ReliableMessaging policy for a given policy set, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **WS-ReliableMessaging**. The "WS-ReliableMessaging settings" on page 2716 form is displayed.
2. Modify one or more of the following properties:

   Standard

   Select the WS-ReliableMessaging specification to use for reliable transmission of your messages. WS-ReliableMessaging Version 1.1 is the default value. Select the WS-ReliableMessaging specification to use for reliable transmission of your messages. WS-ReliableMessaging Version 1.1 is the default value. Details of the supported WS-ReliableMessaging specifications are available at the following web addresses:

   - The WS-ReliableMessaging specification Version 1.0, February 2005.
   - The OASIS WS-ReliableMessaging specification Version 1.1, February 2007.

   Note: If you plan to invoke a .NET-based web service, you must select WS-ReliableMessaging Version 1.0.

   Deliver messages in the order that they were sent

   Select this option if the sender of a request has to receive a response before it sends the next request, or if you want to enable transaction support for inbound (provider) message exchanges as described in "Providing transactional recoverable messaging through WS-ReliableMessaging" on page 3126, or if you want to marginally increase reliability as described in A message is not recovered after a server becomes unavailable, even with the managed persistent quality of service.

   Tip: When you enable this option, WS-ReliableMessaging ensures that the messages are made available to the requester application in the order that they were sent. That is, if WS-ReliableMessaging cannot make a given message available, it will not make any subsequent messages available. However, the requester application must also poll for the messages in the order in which it is to receive them. For example:

   a. WS-ReliableMessaging makes message 1 available, then message 2, then message 3.
   b. The requester application uses asynchronous polling to deliberately poll for message 2, then message 3, then message 1. All three messages are available, so this polling out of order is successful.

   Even though WS-ReliableMessaging is delivering the messages in the order that they were sent, the requester application is choosing to receive them out of order.

Quality of service

Select the radio button for your required quality of service. The three choices are:

Unmanaged non-persistent

You can configure web service applications to use WS-ReliableMessaging with a default in-memory store. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages.

Managed non-persistent

This in-memory quality of service option uses a messaging engine to manage the sequence state, and messages are written to disk if memory is low. This quality of service allows for the re-sending of messages that are lost in the network, and can also recover from server failure. However, state is discarded after a messaging engine restart so in this case you will lose messages.

Managed persistent

This quality of service for asynchronous web service invocations is recoverable. This option also uses a messaging engine and message store to manage the sequence state. Messages are persisted at the web service requester server and at the web service provider server, and are recoverable if the server becomes unavailable. Messages that have not been successfully transmitted when a server becomes unavailable can continue to be transmitted after the server restarts.

The default is unmanaged non-persistent.

Note: All three qualities of service are supported when applications are deployed to the application server. Thin client and client container applications use the first option only.

Note: In WebSphere Application Server Version 6.1, you could also configure whether to use the WS-MakeConnection protocol. This configuration option has now been removed from the administrative console panel, because the product now automatically determines whether WS-MakeConnection is used, based on the following criteria:

- Whether you are using WS-ReliableMessaging Version 1.0 or Version 1.1.
- Whether the requester supports WS-MakeConnection.
- Whether the message exchange protocol is synchronous or asynchronous.

3. Click **OK**.
4. Save your changes to the master configuration.

Results

After you have customized the reliable messaging policy, the associated policy set uses this policy to help ensure reliable delivery of messages.

WS-ReliableMessaging settings:

For the WS-ReliableMessaging policy you can configure the version of the WS-ReliableMessaging standard that you want to use, the order in which messages are delivered, and the required quality of service (the reliability level) for message delivery. The product can enforce these policies on inbound messages and applies them to outbound messages.

To view this page in the console, click the following path: **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **WS-ReliableMessaging**.

With WebSphere Application Server, you can use WS-ReliableMessaging with Java API for XML-Based Web Services (JAX-WS) web services applications that use a SOAP over HTTP binding. Select the

WS-ReliableMessaging specification to use for reliable transmission of your messages. WS-ReliableMessaging Version 1.1 is the default value. Select the WS-ReliableMessaging specification to use for reliable transmission of your messages. WS-ReliableMessaging Version 1.1 is the default value. Details of the supported WS-ReliableMessaging specifications are available at the following web addresses:

- The WS-ReliableMessaging specification Version 1.0, February 2005.
- The OASIS WS-ReliableMessaging specification Version 1.1, February 2007.

Note: If you plan to invoke a .NET-based web service, you must select WS-ReliableMessaging Version 1.0.

Do not edit the policies associated with the provided default policy sets. If you have to modify the reliable messaging policy settings, use a copy of a default policy set or create a new policy set.

At any stage - that is, before or after you have built your reliable web service application, or configured your policy sets - you can set a property that configures endpoints to only support clients that use reliable messaging. This setting is reflected by WS-Policy if engaged.

Standard:

Deliver messages in the order that they were sent:

Select this option if the sender of a request has to receive a response before it sends the next request.

If you enable in-order delivery, you must also ensure that the requester application polls for the messages in the order in which it is to receive them. For more information, see “Configuring the WS-ReliableMessaging policy” on page 2714.

Specifying in-order delivery also marginally increases reliability if you are using the managed persistent quality of service.

Quality of Service: Select one of the following qualities of service:

Unmanaged non-persistent - Tolerates network and remote system failures
You can configure web service applications to use WS-ReliableMessaging with a default in-memory store. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages. The default is Unmanaged Non-Persistent.

Managed non-persistent - Tolerates system, network, and remote system failures, but state is discarded after messaging engine restart
This in-memory quality of service option uses a messaging engine to manage the sequence state, and messages are written to disk if memory is low. This quality of service allows for the re-sending of messages that are lost in the network, and can also recover from server failure. However, state is discarded after a messaging engine restart so in this case you will lose messages.

Managed persistent - Tolerates system, network, and remote system failures
This quality of service for asynchronous web service invocations is recoverable. This option also uses a messaging engine and message store to manage the sequence state. Messages are persisted at the web service requester server and at the web service provider server, and are recoverable if the server becomes unavailable. Messages that have not been successfully transmitted when a server becomes unavailable can continue to be transmitted after the server restarts.

Note:

- All three qualities of service are supported when applications are deployed to the application server. Thin client and client container applications use the first option only.
- For the unmanaged non-persistent quality of service, the messages are stored only in memory. For both of the managed qualities of service, the messages are managed by a messaging engine and stored in a message store. You specify a binding to a bus and messaging engine on the “WS-ReliableMessaging policy binding” form. If your chosen quality of service is Unmanaged Non-Persistent, which does not use a binding to a messaging engine, then any binding that you specify is ignored.

WS-ReliableMessaging policy binding:

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent. The managed qualities of service, managed persistent and managed non-persistent, are supported by the service integration bus. Use this page to select the bus and messaging engine to use for the reliable messaging protocol state.

To view this page in the console, click one of the following paths:

- **Services** > **Policy sets** > **Default policy set bindings** > **Version 6.1 default policy set bindings** > **WS-ReliableMessaging** (This is the default binding for client and provider policy set attachments within WebSphere Application Server Version 6.1 applications, and attachments to service applications that are deployed to a Version 6.1 server.)
- **Services** > **Policy sets** > **General provider policy set bindings** > ***provider_policy_set_binding_name*** > **WS-ReliableMessaging** (This binding is used for the specified provider policy set.)
- **Services** > **Policy sets** > **General client policy set bindings** > ***client_policy_set_binding_name*** > **WS-ReliableMessaging** (This binding is used for the specified client policy set.)

Note:

- You only have to specify a binding to a bus and messaging engine if you are using a managed quality of service. If your chosen quality of service is Unmanaged Non-Persistent, any binding that you specify is ignored. The quality of service is defined on the “WS-ReliableMessaging settings” on page 2716 form for your chosen policy set. For more information, see “Configuring the WS-ReliableMessaging policy” on page 2714.
- When many applications use the same messaging engine, it can impact performance. Factors to consider include the number of applications that are already binding to the messaging engine, the CPU utilization, and the message throughput. To improve performance for a single server configuration, create a new messaging engine to bind to your application.

Bus name:

Specifies a list of available service integration buses in the cell. Use the list to select a bus, or click **Manage buses, bus members, and messaging engines** to add a new bus. The bus that you add is selected for this binding configuration when you return to this panel.

Messaging engine:

Specifies a list of each bus member for the selected bus. Use the list to select a bus member, or click **Manage buses, bus members, and messaging engines** to add a new bus member. The bus member that you add is selected for this binding configuration when you return to this panel.

Configuring the WS-Addressing policy:

When working with policy sets in the administrative console, you can add and configure policies to enable standard addressing of web services.

Before you begin

You can specify policies for custom policy sets. The provided default policy sets cannot be edited. You must create a copy of the default policy set or create a completely new policy set in order to specify the policies for it.

About this task

Adding a WS-Addressing policy enables the support for WS-Addressing. This support provides a standard way to address Web services and include addressing information in messages. Adding a WS-Addressing policy is equivalent to configuring the WSDL file for the web service to specify that WS-Addressing should be used.

To specify or configure the policies associated with a policy set, use the administrative console.

Procedure

1. In the navigation pane of the administrative console, click **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **[Policies] WS-Addressing**. The WS-Addressing settings pane is displayed.
2. Select **WS-Addressing is mandatory** to specify that WS-Addressing information must be included in SOAP message headers. For servers, this setting means that the server returns a fault if it receives a message that does not contain a WS-Addressing header. For clients, this setting means that WS-Addressing headers are always added to SOAP messages. If you have enabled WS-Policy, this requirement is communicated between servers and clients that support WS-Policy.
3. In the **Messaging style** box, select the message exchange pattern to use:
   - **Synchronous and asynchronous**. The targeting of response messages is not restricted.
   - **Synchronous only**. Response messages must be targeted at the WS-Addressing anonymous URI.
   - **Asynchronous only**. Response messages must not be targeted at the WS-Addressing anonymous URI.
4. Click **OK**.
5. Save your changes to the master configuration.

Results

After you have included the WS-Addressing policy in a policy set, the associated policy set uses this policy to address web services.

WS-Addressing policy settings:

Use this page to define the appropriate WS-Addressing policy assertions for this policy set.

To view this administrative console page, click **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **[Policy] WS-Addressing**, when the policy set includes the WS-Addressing policy type.

You can configure the WS-Addressing policy type for both client-side and provider-side policy sets. If you enable WS-Policy, this configuration is communicated between servers and clients that support WS-Policy.

WS-Addressing is mandatory:

Specifies whether a WS-Addressing SOAP header is included on messages.

Data type	Check box
Default	Cleared

Range

Cleared
WS-Addressing is not mandatory. Servers will not generate a fault if they receive a message that does not contain a WS-Addressing header. Clients might not include WS-Addressing headers in SOAP messages, for example, if WS-Policy is enabled and the server does not specify that WS-Addressing is mandatory.

Selected
WS-Addressing is mandatory. Servers return a fault if they receive a message that does not contain a WS-Addressing header. Clients always include WS-Addressing headers in SOAP messages.

Messaging style:

Specifies the messaging style supported by this policy set.

Use the radio buttons to configure the messaging style.

- Select **Synchronous and asynchronous** to specify that there is no restriction on the targeting of response messages.
- Select **Synchronous only** to specify that response messages must be targeted at the WS-Addressing anonymous URI. You might want to use this messaging style in the following situations:
  - The SOAP headers are not signed, and WS-Security is not enabled. Specifying the synchronous message exchange pattern prevents the server sending messages to a third party, thereby preventing the server from potentially taking part in a Denial of Service attack.
  - Clients with a NAT device between themselves and the endpoint. In this configuration, non-anonymous URIs cannot be routed. Use this setting to prevent the client from sending a message containing a ReplyTo endpoint reference with a non-anonymous URI.
- Select **Asynchronous only** to specify that response messages must not be targeted at the WS-Addressing anonymous URI. This setting does not mean that all non-anonymous URIs are supported, therefore a server can return a fault if it receives a response endpoint reference that it cannot process. You might want to use this messaging style if the endpoint has a very long-running invocation time, and you do not want to hold the connection open while waiting for a response.

The following table shows how the messaging style options correspond to WS-Policy assertions.

Table 257. WS-Addressing messaging style and WS-Policy. This table provides a mapping between messaging style options and WS-Policy assertions.

Messaging style	WS-Policy mapping
Synchronous and asynchronous	wsam:AnonymousResponses or wsam:NonAnonymousResponses
Synchronous only	wsam:AnonymousResponses
Asynchronous only	wsam:NonAnonymousResponses

Required No
Data type Radio button

Configuring the HTTP transport policy:

When working with policy sets in the administrative console, you can customize policies to ensure message security. You can customize the Hypertext Transfer Protocol (HTTP) transport policy configuration or use the policy as it is provided with the default settings.

Before you begin

You can configure some settings for default policies for custom policy sets. The provided default policy sets cannot be edited. To customize a policy set, you must create a copy of the default policy set or create a new policy set and specify the policies for the custom policy set.

About this task

You can configure HTTP transport with the HTTP transport policy. HTTP is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol that can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred. HTTP features and HTTP connections properties are applied to outbound messages for both the service client and service provider.

You can only configure a policy through a policy set. Therefore, before you can configure the HTTP transport policy, a policy set must exist that contains the HTTP transport policy. The provided default WSHTTPS policy set is read only and it cannot be edited. To customize a policy set that contains the HTTP transport policy, you must first create a copy of the WSHTTPS default policy set or create a new policy set and add the HTTP transport policy to the new policy set.

Note: The WSHTTPS default policy set contains the HTTP transport policy, the SSL transport policy and WS-Addressing policy. If you do not require the SSL transport policy or the WS-Addressing policy, you can customize your copy of the WSHTTPS default policy set to delete the policies that you do not require.

After you have created a copy of the WSHTTPS default policy set or created a new policy set with the HTTP transport policy added, you can customize the HTTP transport policy. Use the HTTP transport policy settings panel to customize the values of the HTTP transport policy properties such as read or write timeout values. Your customized values for the HTTP transport policy now apply for your policy set that contains that custom HTTP transport policy. You can attach this policy set containing your customized HTTP transport policy to your Java API for XML-Based Web Services (JAX-WS) application, its services, endpoints, or operations. This change affects all JAX-WS applications to which that policy set is attached. To learn more about attaching policy sets to applications, see the documentation for managing policy sets for service providers and service clients at the application level.

For example, if you have multiple policy sets, *mypolicyset1* and *mypolicyset2*, containing the HTTP transport policy, you can customize the HTTP transport policy for each policy set to reflect different properties, such as timeout values. Now, you can attach these customized policy sets to one or more applications and these applications will use the HTTP property values associated with the HTTP transport policy that is contained within the attached policy set.

Procedure

1. Create a policy set that contains the HTTP transport policy.
   a. Create a custom policy set.

      From the administrative console, click **Services > Policy Sets > Application policy sets** . From this panel you can create a new policy set, copy an existing default policy set such as WSHTTPS, import a copy of a policy set from the default repository or you can import an existing policy set from your specified location.
   b. Add the HTTP transport policy to the policy set. From the administrative console, click **Services > Policy Sets > Application policy sets > *policy_set_name***. In the policy collection, click **HTTP transport**. The HTTP transport window displays options for configuring the HTTP settings for the transport policy.

c. In the **Protocol Version** drop down list, click the HTTP version to use. HTTP 1.1 is the default setting but HTTP 1.0 is also available. Selecting HTTP 1.1 enables more of the function on the rest of the HTTP transport window as some of the options are not available for HTTP version 1.0.

d. Complete the HTTP Features section. The following check boxes determine which HTTP features are enabled for this transport:

Session Enabled
Whether the HTTP session is enabled when a message is sent.

Enable chunked transfer encoding
Whether chunked transfer encoding is enabled when a message is sent. This option is only available if HTTP 1.1 is selected in the **Protocol version** field (it is greyed out and disabled if HTTP 1.0 is selected).

Send expect "100-request" header
Displays whether the expect "100-request" header is enabled when a message is sent. This option is only available if HTTP 1.1 is selected in the **Protocol version** field (it is greyed out and disabled if HTTP 1.0 is selected).

Accept URL redirection automatically
Displays whether the URL is automatically redirected when a message is sent.

Compress request content
Displays whether the request content is compressed when a message is sent.

Compress response content
Displays whether the response content is compressed when a message is sent.

e. Complete the HTTP Connections section. The following fields determine how HTTP connections are configured for this transport:

Read timeout
Displays the length of time, in seconds, for the read to time out when a message is sent.

Write timeout
Displays the length of time, in seconds, for the write to time out when a message is sent.

Connection timeout
Displays the length of time, in seconds, for the connection to time out when a message is sent.

Use persistent connection
Displays whether a persistent connection is to be used when a message is sent. This option is only available if HTTP 1.1 is selected in the **Protocol version** field.

Resend enabled
Displays whether or not a message can be resent. Click this check box to enable a message to be sent again.

2. Customize the HTTP transport provider bindings.

a. Navigate to the HTTP transport provider bindings. From the administrative console, click **Services > Policy Sets > General provider policy set bindings > *provider_policy_set_binding_name* > HTTP transport**.

The HTTP transport (bindings) window displays options for configuring the HTTP transport bindings.

b. Specify the properties for the Proxy for outbound asynchronous service responses.

The following fields determine proxy specifications for outbound asynchronous service responses:

Host Displays the host name for the outbound asynchronous service responses proxy.

Port Displays the port number for the outbound asynchronous service responses proxy. You can enter or edit the port number.

User name
Displays the user name for the outbound asynchronous service responses proxy.

Password
Displays a placeholder for the password for the outbound asynchronous service responses proxy. You can enter or edit the password. The actual password is masked.

Confirm password
Displays a placeholder for the password for the outbound asynchronous service responses proxy that must match the one in the **Password** field. The actual password is masked.

c. Specify the properties for the Basic authentication for outbound asynchronous responses.

The following fields determine authentication specifications for outbound asynchronous responses:

User name
Displays the user name for basic authentication of outbound asynchronous responses.

Password
Displays a placeholder for the password for basic authentication of outbound asynchronous responses. The actual password is masked.

Confirm password
Displays a placeholder for the password for basic authentication of outbound asynchronous responses that must match the one in the **Password** field. The actual password is masked.

3. Customize the HTTP transport client bindings.

a. Navigate to the HTTP transport client bindings. From the administrative console, click **Services > Policy Sets > General client policy set bindings > *provider_policy_set_binding_name* > HTTP transport**.

The HTTP transport (bindings) window displays options for configuring the HTTP transport bindings.

b. Specify the properties for the Proxy for outbound service requests. The following fields determine proxy specifications for outbound service requests:

Host Displays the host name for the outbound service request proxy.

Port Displays the port number for the outbound service request proxy.

User name
Displays the user name for the outbound service request proxy.

Password
Displays a placeholder for the password for the outbound service request proxy. The actual password is masked.

Confirm password
Displays a placeholder for the password for the outbound service request proxy that must match the one in the **Password** field. The actual password is masked.

c. Specify the properties for Basic authentication for outbound service requests. The following fields determine authentication specifications for outbound service requests:

User name
Displays the user name for basic authentication of outbound service requests.

Password
Displays a placeholder for the password for basic authentication of outbound service requests. The actual password is masked.

Confirm password
Displays a placeholder for the password for basic authentication of outbound service requests that must match the one in the **Password** field. The actual password is masked.

Results

After you have customized the HTTP transport policy, the associated policy set uses this policy to protect message transmission.

Example

You can attach policy sets to an application, its services, endpoints, or operations. In this example scenario, suppose you have two different JAX-WS service clients for your application, but you want to use different HTTP transport property values for each service client. Specifically, you want to configure a different read or write timeout value for each service client. To modify the HTTP timeout values, you can edit the values of the HTTP transport policy that is contained within the policy set that is attached to your application or in this case, your service client. This change affects all applications to which the policy set containing the custom HTTP transport policy is attached.

This example describes the steps for configuring different read, write, and connection timeout values for service clients deployed in the same application server. This example makes the following assumptions:

- There are two JAX-WS service clients, ServiceClient1 and ServiceClient2, that are deployed in the application server.
- The HTTP transport policy has not been previously attached to these applications.

1. Create two new policy sets and add the HTTP transport policy to them. For example: *HTTPServiceClient1Policy* and *HTTPServiceClient2Policy*
   a. Click **Services > Policy sets > Application policy sets > New** .
   b. Enter the name of the new application policy set, *HTTPServiceClient1Policy*.
   c. From the Policies collection, click **Add > HTTP transport**.
   d. Click **Apply** and **Save** to save your changes to the master configuration.
   e. Repeat these steps to create the HTTPServiceClient2Policy.
2. Customize the HTTP transport policy settings for the newly created HTTPServiceClient1Policy and HTTPServiceClient2Policy policy sets. For example, customize the read and write timeout values for the HTTP transport policy contained in the HTTPServiceClient1Policy policy set and the connection timeout value for the HTTP transport policy contained in the HTTPServiceClient2Policy policy set.
   a. Click **Services > Policy sets > Application policy sets >***HTTPServiceClient1Policy* .
   b. From the Policies collection, click **HTTP transport**.
   c. From the HTTP transport policy configuration panel, change the HTTP connection read and write timeout values to 500 seconds.
   d. Click **Apply** and **Save** to save your changes to the master configuration.
   e. Click **Services > Policy sets > Application policy sets >***HTTPServiceClient2Policy* .
   f. From the Policies collection, click **HTTP transport**.
   g. From the HTTP transport policy configuration panel, change the HTTP connection timeout value to 360 seconds.
   h. Click **Apply** and **Save** to save your changes to the master configuration.
3. Attach the custom HTTP transport policy, HTTPServiceClient1Policy, to your application, ServiceClient1. Similarly, attach the custom HTTP transport policy, HTTPServiceClient2Policy, to ServiceClient2.
   a. Click **Services > Service clients >** *ServiceClient1*.
   b. From the Policy set attachments collection, select the service, *ServiceClient1*.
   c. Click **Attach Client Policy Set** and click on *HTTPServiceClient1Policy*.
   d. Click **Save** to save your changes to the master configuration.
   e. Click **Services > Service clients >** *ServiceClient2*.
   f. From the Policy set attachments collection, select the service, *ServiceClient2*.

g. Click **Attach Client Policy Set** and click on *HTTPServiceClient2Policy*.
h. Click **Save** to save your changes to the master configuration.

As a result, the ServiceClient1 application now has the HTTPServiceClient1Policy attached and the HTTP sessions will use a read and write timeout value of 500 seconds. The ServiceClient2 application has the HTTPServiceClient2Policy attached and the HTTP sessions will use a connection timeout value of 360 seconds.

What to do next

You can customize policies to ensure message security by configuring the SSL transport policy.

HTTP transport policy settings:

Use this page to define HTTP transport policy configuration. HTTP features and HTTP connection policies are applied to outbound messages. Any changes to the HTTP transport policy from this console page affects all Java API for XML-Based Web Services (JAX-WS) applications to which this custom HTTP transport policy is attached.

To view this administrative console page, click **Services > Policy sets > Application policy sets** >*policy_set_name* > **HTTP transport**, where *policy_set_name*, applies to any policy set that contains HTTP transport policy.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

You can only configure a policy through a policy set. Therefore, before you can configure the HTTP transport policy, a policy set must exist that contains the HTTP transport policy.

The WSHTTPS default policy set is provided with the application server and it contains the HTTP transport policy, the SSL transport policy and the WS-Addressing policy. The provided default WSHTTPS policy set is read only and it cannot be edited. To customize a policy set that contains the HTTP transport policy, you must first create a copy of the WSHTTPS default policy set or create a new policy set and add the HTTP transport policy to the new policy set.

After you customize values for the HTTP transport policy, these values now apply for your policy set that contains that custom HTTP transport policy. You can attach this policy set that contains your customized HTTP transport policy to your application, its services, endpoints, or operations. This change affects all JAX-WS applications to which that policy set is attached. To learn more about attaching policy sets to applications, see the documentation for managing policy sets for service providers and service clients at the application level.

Protocol version:

Specifies the version of HTTP protocol to use. Use this list to specify the version of HTTP protocol. The default value is `HTTP 1.1`. The `HTTP 1.0` value is also a valid option.

Some of the remaining options on the HTTP Transport panel only work with HTTP Version 1.1. The following brief descriptions compare these options:

HTTP 1.0
Allows messages to be in MIME-like format, containing meta information about the data transferred and modifiers on the request, response, or both. However, HTTP 1.0 does not sufficiently address the effects of hierarchical proxies, caching, the need for persistent connections, or virtual hosts.

HTTP 1.1
Enables each of two communicating applications to determine the true capabilities or the other. This protocol includes more stringent requirements than HTTP 1.0 to ensure reliable implementation of features.

Session enabled:

Specifies whether the HTTP session is enabled when a message is sent. Select this check box to enable an HTTP session.

If this property is used within a policy set that is attached to a service client, then it indicates whether HTTP session information is propagated to subsequent requests invoked by the same client application. If the property is enabled, the HTTP session information is returned to the service client in a response message is sent in subsequent requests invoked using the same RequestContext object.

If this property is used within a policy set that is attached to a service provider, then it indicates whether or not a new HTTP session is created when a request is being processed. If the property is enabled, then as a request is being processed, a new HTTP session is created if one does not already exist. This HTTP session information is then returned to the service client in the response message.

Enable chunked transfer encoding:

Specifies whether chunked transfer encoding is enabled when a message is sent. Select this check box to enable a chunked transfer encoding. This option is only available if you select HTTP 1.1 in the **Protocol version** field. This option is disabled if you selected the HTTP 1.0 protocol.

The default for this property is `true`.

Send expect "100-request" header:

Specifies whether the `expect "100-request"` header is enabled when a message is sent. Select this check box to enable the `expect "100-request"` header. This option is only available if you selected HTTP 1.1 in the **Protocol version** field. This option is disabled if you selected the HTTP 1.0 protocol.

The purpose of the 100 status is to allow a client that is sending a request message with a request body to determine if the origin server accepts the request, based on the request headers, before the client sends the request body. In some cases, you might not want the client to send the body if the server rejects the message without looking at the body.

The Expect request-header field is used to indicate that particular server behaviors are required by the client. A server that cannot comply with any of the expectation values in the Expect field if a request responds with an appropriate error status.

Accept URL redirection automatically:

Specifies whether the automatic URL redirection is accepted when a message is sent. Select this check box to enable a URL that has been automatically redirected to be accepted.

Compress request content:

Specifies whether the request content is compressed when a message is sent. Content coding is used to allow a document to be compressed without losing the identity of its underlying media type and without loss of information. Select this check box to enable request content that you want to compress. Clicking the Compress request content button enables the **Compression format** option to select the compression method. The default value for the compression format is `gzip`.

Compress response content:

Specifies whether the response content is compressed when a message is sent. Content coding is used to allow a document to be compressed without losing the identity of its underlying media type and without loss of information. Select this check box to enable response content that you want to compress. Clicking the Compress response content button enables the **Compression format** option as he compression method. The default value for the compression format is `gzip`.

Read timeout:

Specifies the length of time, in seconds, for the Web services client to completely read the SOAP response. If the read process does not complete within the specified time, a SOAP fault error is generated on the client machine.

Write timeout:

Specifies the length of time, in seconds, for the write action to time out when a message is sent. Specify the time, in seconds, to enable the write to time out length of time.

Connection timeout:

Specifies the length of time, in seconds, for the connection to time out when a message is sent. Specify the time, in seconds, to enable the connection to time out length of time.

Use persistent connection:

Specifies whether a persistent connection is used when a message is sent. Select this check box to enable use of a persistent connection. This option is only available if you selected HTTP 1.1 in the **Protocol version** field. This option is disabled if you selected the HTTP 1.0 protocol.

Resend enabled:

Specifies whether a message can be resent. Select this check box to resend a message.

HTTP transport bindings settings:

Use this page to define the HTTP transport bindings for the HTTP transport policy.

To configure the HTTP transport bindings for the HTTP transport policy, perform the following:

1. Navigate to the general bindings collection panel using either the **Services > Policy sets > General client policy set bindings** or **Services > Policy sets > General provider policy set bindings** path.
2. Click a general binding in the Name column.
3. Click the **HTTP transport** policy in the Policies table.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Important: You can also configure HTTP transport properties such as read or write timeout values for JAX-WS applications that are deployed in the same application server. If you want to customize these HTTP properties, you must edit the HTTP transport policy. To customize the HTTP transport policy settings, click **Services > Policy sets > Application policy sets >*policy_set_name*>HTTP transport policy** where *policy_set_name* applies to any policy set that contains the HTTP transport policy. Your customized values for the HTTP transport policy now apply for your policy set that contains that custom HTTP transport policy. You can attach this policy set that contains your customized HTTP transport policy to your application, its services, endpoints, or operations. This change affects all JAX-WS applications to which that

policy set is attached. To learn more about attaching policy sets to applications, see the documentation for managing policy sets for service providers and service clients at the application level.

Proxy for outbound service requests – Host:

Specifies the host name for the outbound service request proxy. In this field, you can change the name of the host if you are editing an existing set of application specific bindings or you can enter the host name for the outbound service request proxy.

Proxy for outbound service requests – Port:

Specifies the port number for the outbound service request proxy. You can edit an existing port number, or enter a new port number for the outbound service request proxy in this field.

Proxy for outbound service requests – User name:

Specifies the user name for the outbound service request proxy. Enter a user name in this field.

Proxy for outbound service requests – Password:

Specifies a placeholder for the outbound service request proxy password. The actual password is masked.

Proxy for outbound service requests – Confirm password:

Specifies a placeholder for the outbound service request proxy password. Re-enter the password you entered in the **Password** field. The actual password is masked.

Basic authentication for outbound service requests – User name:

Specifies the user name for basic authentication of outbound service requests. Enter or edit the user name.

Basic authentication for outbound service requests – Password:

Specifies a placeholder for basic authentication of outbound service requests password. The actual password is masked.

Basic authentication for outbound service requests – Confirm password:

Specifies a placeholder for basic authentication of outbound service requests password. Re-enter the same password as in the **Password** field. The actual password is masked.

Proxy for outbound asynchronous service responses – Host:

Specifies the host name for the outbound asynchronous service responses proxy. You can enter or edit the host name.

Proxy for outbound asynchronous service responses – Port:

Specifies the port number for the outbound asynchronous service responses proxy. You can enter or edit the port number.

Proxy for outbound asynchronous service responses – User name:

Specifies the user name for the outbound asynchronous service responses proxy. You can enter or edit the user name.

Proxy for outbound asynchronous service responses – Password:

Specifies a placeholder for the outbound asynchronous service responses proxy password. You can enter or edit the password. The actual password is masked.

Proxy for outbound asynchronous service responses – Confirm password:

Specifies a placeholder for the outbound asynchronous service responses proxy password. You can re-enter or edit the password. The actual password is masked.

Basic authentication for outbound asynchronous service responses – User name:

Specifies the user name for basic authentication of outbound asynchronous responses. You can enter or edit the user name.

Basic authentication for outbound asynchronous service responses – Password:

Specifies a placeholder for basic authentication of outbound asynchronous responses password. You can enter or edit the password in this field. The actual password is masked.

Basic authentication for outbound asynchronous service responses – Confirm password:

Specifies a placeholder for basic authentication of outbound asynchronous responses password. Re-enter the password in this field. The actual password is masked.

Custom Properties – Name:

Specifies the name of custom property. Custom properties are not initially displayed in this column until you define them.

Custom Properties – Value:

Specifies the value of the custom property. With the **Value** entry field, you can enter, edit, or delete the value for a custom property.

Click one of the following buttons to enable the action described:

Button	Resulting Action
New	Creates a new custom property entry. To add a custom property, enter the name and value.
Delete	Removes the selected custom property.
Edit	You can edit a selected custom property. It is only displayed when one or more properties exist.

Configuring the Java Message Service (JMS) transport policy:

You can define a Java Message Service (JMS) transport policy configuration if you are using SOAP over JMS with your Java API for XML-Based Web Services (JAX-WS) applications.

Before you begin

You can configure some settings for policies for custom policy sets. The provided default policy sets cannot be edited. You must create a copy of the default policy set or create a new policy set in order to specify the policies for it.

About this task

When using the SOAP over JMS transport with JAX-WS applications, you can customize the transport by configuring the JMS transport policy. The SOAP over JMS transport provides an alternative to HTTPS for transporting SOAP requests and response messages between clients and servers. See the documentation on using SOAP over JMS to transport web services to learn more about this transport protocol.

You can only configure a policy through a policy set. Therefore, before you can configure the JMS transport policy, a policy set must exist that contains the JMS transport policy. To customize a policy set that contains the JMS transport policy, you must first create a policy set and add the JMS transport policy to the new policy set.

Use the JMS transport policy settings panel to customize the values of the JMS transport policy properties, such as the request timeout value. Your customized values for the JMS transport policy now apply for your policy set that contains that custom JMS transport policy. You can attach this policy set containing your customized JMS transport policy to your JAX-WS application, its services, endpoints, or operations. This change affects all JAX-WS applications to which that policy set is attached. To learn more about attaching policy sets to applications, see the documentation for managing policy sets for service providers and service clients at the application level.

Procedure

1. Create a policy set that contains the JMS transport policy.
   a. Create a custom policy set. From the administrative console, click **Services > Policy Sets > Application policy sets**. From this panel you can create a new policy set, import a copy of a policy set from the default repository, or you can import an existing policy set from your specified location.
   b. Add the JMS transport policy to the policy set. From the administrative console, click **Services > Policy Sets > Application policy sets > *policy_set_name***. In the policy collection, click **JMS transport**. The JMS transport window displays options for configuring the JMS settings for the transport policy.
   c. Specify the JMS connection properties for the JMS transport requests. The following fields configure the JMS features for this transport:

      Request timeout
      Specifies whether to enable a request timeout value. The request timeout value is the amount of time that the client waits for a response after sending the request to the server. The range is from 0 to 2147483647.

      Allow transactional messaging for one-way and asynchronous operations
      Specifies to enable a client to use transactions in one-way or asynchronous two-way requests. Select this check box to enable transactional messaging.

      When this option is selected, the client runtime environment exchanges SOAP request and response messages with the server over the JMS transport in a transactional manner if the client is operating under a transaction. This process indicates that the clients transaction is used to send the SOAP request message to the destination queue or topic, and the server receives the request message only after the client commits the transaction. Similarly, the server receives the request message under the control of a container-managed transaction and sends the reply message, if applicable, back to the client using that same transaction. The client then receives the reply message only after the server transaction has been committed.

      If this option is not selected, then the client and server runtime environments perform messaging operations in a non-transactional manner as transactions are temporarily suspended for the JMS request. The transactions are enabled again after the request has completed.

Note: Transactional messaging operations are not supported for two-way synchronous operations as this leads to a deadlock condition.

2. Customize the JMS transport provider bindings.
   a. Navigate to the JMS transport provider bindings. From the administrative console, click **Services > Policy Sets > General provider policy set bindings > *provider_policy_set_binding_name* > JMS transport**.

      The JMS transport provider bindings window displays options for defining basic authentication for asynchronous service responses and custom properties for the JMS service provider binding configuration.

   b. Specify the properties for basic authentication for asynchronous service responses.

      You can use the JMS transport provider policy bindings to configure a service that uses the JMS transport to send asynchronous response messages back to the client. The application server runtime environment uses the user name and password that you configure when connecting to the JMS messaging provider and this configuration enables the service to send an asynchronous response message to the client in a secure manner.

      The following fields determine the authentication requirements for responses from the server:

      User name
      Specifies the user name for the asynchronous service responses for the service provider.

      Password
      Specifies a placeholder for the password for the asynchronous service responses from the service provider. You can enter or edit the password in this field. The actual password is masked.

      Confirm password
      Specifies a placeholder for the password for the asynchronous service responses from the service provider that must match the one in the **Password** field. The actual password is masked.

3. Customize the JMS transport client bindings.
   a. Navigate to the JMS transport client bindings. From the administrative console, click **Services > Policy Sets > General client policy set bindings > *client_policy_set_binding_name* > JMS transport**.

      The JMS transport provider bindings window displays options for defining basic authentication for outbound service requests and custom properties for the JMS client binding configuration.

   b. Specify the Basic Authentication for Outbound Service Requests properties.

      You can use the JMS transport client policy bindings to configure a client that uses the JMS transport to send a request message to the server. The client runtime environment uses the user name and password that you configure when connecting to the JMS messaging provide. This configuration enables the client to send the request message to the server in a secure manner.

      The following fields determine the authentication requirements for requests sent to the server:

      User name
      Specifies the user name that is used by the client runtime when connecting to the JMS messaging provider to send an outbound request to the destination queue or topic. Enter a user name in this field.

      Password
      Specifies a placeholder for the password that is used by the client runtime when connecting to the JMS messaging provider to send an outbound request to the destination queue or topic. You can enter or edit the password in this field. The actual password is masked.

      Confirm password
      Specifies a placeholder for the password that is used by the client runtime when connecting to the JMS messaging provider to send an outbound request to the destination

queue or topic. Re-enter the password in this field. This password must match the one in the **Password** field. The actual password is masked.

Results

After you have customized the JMS transport policy, the associated policy set uses this policy to configure the runtime behavior of the SOAP over JMS transport.

Example

You can attach policy sets to an application, its services, endpoints, or operations. In this example scenario, suppose you have two different JAX-WS service clients for your application, but you want to use different JMS transport request timeout values for each service client. To modify the JMS request timeout values, you can edit the values of the JMS transport policy that is contained within the policy set that is attached to your application or in this case, your service client. This change affects all applications to which the policy set containing the custom JMS transport policy is attached.

This example describes the steps for configuring different request timeout values for service clients deployed in the same application server. This example includes the following assumptions:

- Two JAX-WS service clients exist, ServiceClient1 and ServiceClient2, that are deployed in the application server.
- The JMS transport policy has not been previously attached to these applications.

1. Create two new policy sets and add the JMS transport policy to them. For example: *JMSServiceClient1Policy* and *JMSServiceClient2Policy*
   a. Click **Services > Policy sets > Application policy sets > New** .
   b. Enter the name of the new application policy set, `JMSServiceClient1Policy`.
   c. From the Policies collection, click **Add > JMS transport**.
   d. Click **Apply** and **Save** to save your changes to the master configuration.
   e. Repeat these steps to create the JMSServiceClient2Policy.
2. Customize the JMS transport policy settings for the newly created JMSServiceClient1Policy and JMSServiceClient2Policy policy sets. For example, set the request timeout value to `180` seconds for the JMS transport policy contained in the JMSServiceClient1Policy. The JMS transport policy contained in the JMSServiceClient2Policy specifies `300` seconds as the request timeout value.
   a. Click **Services > Policy sets > Application policy sets > *JMSServiceClient1Policy*** .
   b. From the Policies collection, click **JMS transport**.
   c. From the JMS transport policy configuration panel, specify `180` seconds for the request timeout value.
   d. Click **Apply** and **Save** to save your changes to the master configuration.
   e. Click **Services > Policy sets > Application policy sets > *JMSServiceClient2Policy*** .
   f. From the Policies collection, click **JMS transport**.
   g. From the JMS transport policy configuration panel, specify 300 seconds for the request timeout value.
   h. Click **Apply** and **Save** to save your changes to the master configuration.
3. Attach the custom JMS transport policy, JMSServiceClient1Policy, to your application, ServiceClient1. Similarly, attach the custom JMS transport policy, JMSServiceClient2Policy, to ServiceClient2.
   a. Click **Services > Service clients >** *ServiceClient1*.
   b. From the Policy set attachments collection, select the service, *ServiceClient1*.
   c. Click **Attach Client Policy Set**, and click *JMSServiceClient1Policy*.
   d. Click **Save** to save your changes to the master configuration.
   e. Click **Services > Service clients >** *ServiceClient2*.

f. From the Policy set attachments collection, select the service, *ServiceClient1*.
g. Click **Attach Client Policy Set**, and click *JMSServiceClient2Policy*.
h. Click **Save** to save your changes to the master configuration.

As a result, the ServiceClient1 application now has the JMSServiceClient1Policy attached, and the JMS sessions use a request timeout of 180 seconds. The ServiceClient2 application has the policy, JMSServiceClient2Policy, attached and the JMS sessions use a request timeout of 300 seconds.

What to do next

You can customize other policies that you might need for your application.

JMS transport policy settings:

Use this page to configure settings for the Java Message Service (JMS) transport policy. You can configure a client that is using the JMS transport policy to exchange request and response messages with the server.

To view this administrative console page:
1. Click **Services**, expand **Policy sets**, and click **Application policy sets**.
2. Click **New** to create a new application policy set and provide a name for it in the **Name** field.
3. Under the **Policy** heading, click **Add** and select **JMS transport**.
4. Click **OK** or **Apply**.
5. Click **JMS transport** to view the JMS transport policy settings panel.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

You can only configure a policy through a policy set. Therefore, before you can configure the JMS transport policy, a policy set must exist that contains the JMS transport policy.

To customize a policy set that contains the JMS transport policy, you must create a new policy set, import a copy of a policy set from the default repository, or you can import an existing policy set from your specified location. After you have an editable policy set, you can add the JMS transport policy to your policy set.

After you customize values for the JMS transport policy, these values now apply for your policy set that contains that customized JMS transport policy. You can attach this policy set that contains your customized JMS transport policy to your application, its services, endpoints, or operations. This change affects all JAX-WS applications to which that policy set is attached. To learn more about attaching policy sets to applications, read about managing policy sets for service providers and service clients at the application level.

Client JMS connection properties - Request timeout:

Specifies the request timeout value. The request timeout value is the amount of time that the client waits for a response after sending the request to the server. The default value is 300 seconds.

Client JMS connection properties - Allow transactional messaging for one-way and asynchronous operations:

Specifies to enable a client to use transactions in one-way or asynchronous two-way requests. Select this check box to enable transactional messaging.

If this option is selected, the client runtime exchanges SOAP request and response messages with the server over the JMS transport in a transactional manner if the client is operating under a transaction. Therefore, the client transaction is used to send the SOAP request message to the destination queue or topic, and the server receives the request message only after the client commits the transaction. Similarly, the server receives the request message under the control of a container-managed transaction and sends the reply message, if applicable, back to the client using that same transaction. The client then receives the reply message only after the server transaction is committed.

If this option is not selected, the client and server runtimes perform messaging operations in a non-transactional manner.

Note: Transactional messaging operations are not supported for two-way synchronous operations as this leads to a deadlock condition.

.

JMS transport bindings:

Use this page to define the Java Message Service (JMS) transport provider or client bindings configuration.

If you are using JMS transport provider bindings, to view this administrative console page, complete the following actions:

1. Click **Services > Policy Sets > General provider policy set bindings**.
2. Click ***provider_policy_set_binding_name***.
3. In the policy collection, click **JMS transport**.

If the JMS transport policy has not been added to the selected general provider policy set, then use the create general provider bindings panel to add the JMS transport policy to the selected general provider policy set.

You can use the JMS transport provider policy bindings to configure a service that uses the JMS transport to send asynchronous response messages back to the client. The application server run time uses the user name and password that you configure when connecting to the JMS messaging provider and this configuration enables the service to send an asynchronous response message to the client in a secure manner.

If you are using JMS transport client bindings, to view this administrative console page, complete the following actions:

1. Click **Services > Policy Sets > General client policy set bindings**.
2. Click ***client_policy_set_binding_name***.
3. In the policy collection, click **JMS transport**.

If the JMS transport policy has not been added to the selected general client policy set, then use the create general client bindings panel to add the JMS transport policy to the selected general client policy set.

You can use the JMS transport client policy bindings to configure a client that uses the JMS transport to send a request message to the server. The client run time uses the user name and password that you specify when connecting to the JMS messaging provider, and this configuration enables the client to send the request message to the server in a secure manner.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Important: You can also configure JMS transport properties, such as request timeout values or whether to enable transactional messaging for one-way asynchronous operations for JAX-WS

applications that are deployed in the same application server. If you want to customize these JMS properties, you must edit the JMS transport policy. To customize the JMS transport policy settings, click **Services > Policy sets > Application policy sets > *policy_set_name* >JMS transport policy**, where *policy_set_name* applies to any policy set that contains the JMS transport policy. Your customized values for the JMS transport policy now apply for your policy set that contains that custom JMS transport policy. You can attach this policy set that contains your customized JMS transport policy to your application, its services, endpoints, or operations. This change affects all JAX-WS applications to which that policy set is attached. To learn more about attaching policy sets to applications, see the documentation for managing policy sets for service providers and service clients at the application level.

Basic Authentication – User name:

For the service provider, this field specifies the user name for the asynchronous service responses. For the client, this field specifies the user name that is used by the client runtime when connecting to the JMS messaging provider to send an outbound request to the destination queue or topic. Enter a user name in this field.

Basic Authentication – Password:

For the service provider, this field specifies a placeholder for the password of the asynchronous service responses. For the client, this field specifies the password that is used by the client runtime when connecting to the JMS messaging provider to send an outbound request to the destination queue or topic. You can enter or edit the password in this field. The actual password is masked.

Basic Authentication – Confirm password:

For the service provider, this field specifies a placeholder for the password for the asynchronous service responses. For the client, this field specifies the password that is used by the client runtime when connecting to the JMS messaging provider to send an outbound request to the destination queue or topic. Re-enter the password in this field. The actual password is masked.

Custom Properties – Name:

Specifies the name of custom property. Custom properties are not initially displayed in this column until you define them.

Click one of the following buttons to enable the described action:

Button	Resulting Action
New	Creates a new custom property entry. To add a custom property, enter the name and value.
Delete	Removes the selected custom property.
Edit	Edit a selected custom property. This button is only displayed when one or more properties exist.

Custom Properties – Value:

Specifies the value of the custom property. With the **Value** entry field, you can enter, edit, or delete the value for a custom property.

Configuring the WS-Transaction policy:

When you work with policy sets in the administrative console, you can configure the WS-Transaction policy type for the WS-AtomicTransaction (WS-AT) and the WS-BusinessActivity (WS-BA) protocols. You can configure whether a client propagates, and a server receives, a WS-AT context, and whether a client propagates, and a server receives, a WS-BA context.

Before you begin

You must be working with a policy set that includes the WS-Transaction policy type.

Do not edit the policies associated with the provided default policy sets. To modify the WS-Transaction policy settings, use a copy of a default policy set or create a new policy set.

About this task

You can configure the policies for the WS-AtomicTransaction and WS-BusinessActivity protocols. The WS-AT protocol supports coordination of activities so that either all the activities occur, or none of them occur. The WS-BA protocol supports coordination of activities that are more loosely coupled than atomic transactions and that therefore require a compensation process if a failure occurs in the business activity.

When you add a WS-Transaction policy, it is equivalent to setting the following deployment descriptors that are associated with an EJB or web module:

- Use Web Services Atomic Transaction
- Send Web Services Atomic Transaction on requests
- Execute using Web Services Atomic Transaction on incoming requests

A WS-BA context is sent if the client is running in a BusinessActivity scope (BAScope). A provider runs in a BAScope if it receives a message that contains a WS-BA context, as long as the provider is set to run Enterprise JavaBeans (EJB) methods in a Business Activity scope.

Procedure

1. In the navigation pane of the administrative console, click **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **[Policies] WS-Transaction**. The WS-Transactions settings pane is displayed.
2. In the WS-AtomicTransaction section, select the option you require:
   - **Mandatory**. For a client, the client always propagates a WS-AT context on an outbound request. For a server, any request that is received must include a WS-AT context, otherwise the request is rejected.
   - **Supports**. For a client, the client can propagate a WS-AT context on an outbound request when that context is available. For a server, if a request includes a WS-AT context, the context is imported and established on the thread before the request is processed.
   - **Never**. For a client, the client never propagates a WS-AT context on an outbound request. For a server, any request that is received must not include a WS-AT context, otherwise the request is rejected.
3. In the WS-BusinessActivity section, select the option you require:
   - **Mandatory**. For a client, the client always propagates a WS-BA context on an outbound request. For a server, any request that is received must include a WS-BA context, otherwise the request is rejected.
   - **Supports**. For a client, the client can propagate a WS-BA context on an outbound request when that context is available. For a server, if a request includes a WS-BA context, the context is imported and established on the thread before the request is processed.
   - **Never**. For a client, the client never propagates a WS-BA context on an outbound request. For a server, any request that is received must not include a WS-BA context, otherwise the request is rejected.

4. Click **OK**.
5. Save your changes to the master configuration.

Results

After you configure the WS-Transaction policy, the associated policy set uses this policy to support WS-AtomicTransaction and WS-BusinessActivity.

WS-Transaction policy settings:

Use this page to specify the policies for the WS-AtomicTransaction (WS-AT) and WS-BusinessActivity (WS-BA) protocols. WS-AT supports coordination of activities so that either all the activities occur, or none of them occur. WS-BA supports coordination of activities that are more loosely coupled than atomic transactions, and that therefore, require a compensation process if an error occurs.

To view this page in the console, click the following path: **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **[Policy] WS-Transaction**, when the policy set includes the WS-Transaction policy type.

You can configure the WS-Transaction policy type for both client and provider policy sets.

WS-AtomicTransaction: Specifies behavior with the WS-AT policy. The options are:

Mandatory
For a client, the client always propagates a WS-AT context on an outbound request. If there is no transaction on the thread when the request is made, the attempt to make the request fails.

For a server, any request that is received must include a WS-AT context, otherwise the request is rejected. If any Web Services Description Language (WSDL) is generated for the web service with which the policy type is associated, a policy assertion is included that indicates that an operation must be invoked with an atomic transaction context.

Supports
For a client, the client can propagate a WS-AT context on an outbound request when it is available. For example, a transaction is associated with the thread that makes the request, and the policy of the provider requires WS-AT context.

For a server, if a request includes a WS-AT context, the context is imported and established on the thread before the request is processed. If a request does not include a WS-AT context, the request is processed as usual. If any WSDL is generated for the web service with which the policy type is associated, a policy assertion is included that indicates that an operation supports invocation with an atomic transaction context when that context is available.

Never For a client, the client never propagates a WS-AT context on an outbound request.

For a server, any request that is received must not include a WS-AT context, otherwise the request is rejected with a MustUnderstand error. If any WSDL is generated for the web service with which the policy type is associated, that WSDL does not include a policy assertion for an atomic transaction context.

WS-BusinessActivity: Specifies behavior with the WS-BA policy. The options are:

Mandatory
For a client, the client always propagates a WS-BA context on an outbound request. If there is no business activity scope on the thread when the request is made, the attempt to make the request fails.

For a server, any request that is received must include a WS-BA context, otherwise the request is rejected. If any WSDL is generated for the web service with which the policy type is associated, a policy assertion is included that indicates that an operation must be invoked with a business activity context.

Supports
For a client, the client can propagate a WS-BA context on an outbound request when it is available. For example, a business activity scope is associated with the thread that makes the request, and the policy of the provider requires a WS-BA context.

For a server, if a request includes a WS-BA context, the context is imported and established on the thread before the request is processed. If a request does not include a WS-BA context, the request is processed as usual. If any WSDL is generated for the web service with which the policy type is associated, a policy assertion is included that indicates that an operation supports invocation with a business activity context when that context is available.

Never For a client, the client never propagates a WS-BA context on an outbound request.

For a server, any request that is received must not include a WS-BA context, otherwise the request is rejected with a MustUnderstand error. If any WSDL is generated for the web service with which the policy type is associated, that WSDL does not include a policy assertion for a business activity context.

Configuring the WS-Security policy:

When working with policy sets in the administrative console, you can customize policies to ensure message security. The WS-Security policy can be configured to apply a message security (WS-Security) profile to requests. Message security policies are applied to requests and enforced on responses to support interoperability.

Before you begin

You can configure some settings for default policies for custom policy sets. The provided default policy sets cannot be edited. You must create a copy of the default policy set or create a completely new policy set in order to specify the policies for it.

About this task

Message security policies are applied to requests and enforced on responses to support interoperability.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. Use the WS-Security policy panel to begin configuring the WS-Security policy. To access the WS-Security policy panel, from the administrative console, click **Services > Policy sets > Application policy sets > *policy_set_name* > WS-Security policy**.
2. Choose which type of message security to configure.
   - Click the Main policy link to specify how message security policies are applied to requests and enforced on responses to support interoperability.
   - Click the Bootstrap policy link to configure how secure conversations are established. A bootstrap policy might already be configured. If no bootstrap policy is currently configured, first ensure that you have enabled message security with symmetric signature and encryption policies and secure conversation tokens for both integrity and confidentiality protection.
3. Use the Main policy settings panel or the Bootstrap policy settings panel to specify how message security policies are applied to requests and enforced on responses. Assertions for WS-Security

versions are already generated based on assertions in the policy set. If the policy set includes a WS-S 1.1 assertion, then WS-S 1.1 itself is asserted. Configure the settings on this panel to configure main or bootstrap policy settings:

a. Select whether Message level protection is required. Select this check box if any of the message parts should be digitally signed or encrypted or if a timestamp should be inserted in the message. It this box is unchecked, the Signature confirmation, Key symmetry, and Timestamp and Security header layout options are disabled.

b. Specify whether signature confirmation is required. Click this check box to require signature confirmation.

c. Configure the settings in the Key Symmetry section. The following fields can be configured in the Key symmetry section:

   Use symmetric tokens
   Click this radio button to use symmetric tokens. You can then configure symmetric tokens with the **Symmetric signature and encryption policies** link. Click this link to access the Symmetric Signature and Encryption Policies panel where you can create the trust context in which to use symmetric tokens. Using the same token for signing and validating messages and encrypting and decrypting messages provides better performance than can be achieved with asymmetric tokens. Symmetric tokens should be used within a trust context.

   Use asymmetric tokens
   Click this link to access the Asymmetric Signature and Encryption Policies panel where you can create the trust context (message integrity and confidentiality) in which to use asymmetric tokens. You can do this by specifying which token type to use for the initiator and recipient signature as well as the initiator and recipient encryption.

   Include timestamp in header
   Click this check box to include a timestamp in the header. You can then specify if the timestamp is positioned first or last in the header by using the Security header layout radio button options:
   - **Strict: Declarations must precede use**
   - **Layout (Lax): Order of contents can vary**
   - **Lax but timestamp required first in header**
   - **Lax but timestamp required last in header**

d. Optional: Click the **Algorithms** link under the **Policy Details** section if you want to access the Algorithms panel to view and select from available algorithms. The available algorithms include cryptographic algorithms and their key lengths, as well as canonicalization algorithms for reconciling XML differences. Click this link to view the cryptographic and cannonicalization algorithms that are supported.

e. Optional: Configure the request settings. Click either of the following links to configure request settings:

   Request message part protection
   Links to configuration for request message part protection. Click this link to define which message parts are to be protected and how that protection is provided.

   Request token policies
   Links to configuration for request token policies. Click this link to define policies that specify which types of security tokens are supported and the properties of those token types.

f. Optional: Configure the response settings. Click either of the following links to configure response settings:

   Response message part protection
   Links to configuration for response message part protection. Click this link to define which message parts are to be protected and how that protection is provided.

Response token policies

Links to configuration for response token policies. Click this link to define policies that specify which types of security tokens are supported and the properties of those token types.

Results

Once you have customized the WS-Security policy, the associated policy set uses this policy to protect messages.

WS-Security policy settings:

Use this page to configure the WS-Security policy and apply a message security WS-Security profile to requests. WS-Security policies are applied to requests and enforced on responses to support inter-operability.

To view this administrative console page, click **Services > Policy sets > Application policy sets > *policy_set_name* > WS-Security policy**.

Main policy:

Links to configuration settings for a main policy. Use the Main policy link if you want to configure how WS-Security policies are applied to requests and enforced on responses to support interoperability.

Secure conversation bootstrap policy:

Links to configuration settings for a secure conversation bootstrap policy. Use the bootstrap policy link if you want to configure how secure conversations are established. To configure a secure conversation bootstrap policy, first ensure that you have enabled message security in the main policy with symmetric signature and encryption policies and secure conversation tokens for both integrity and confidentiality protection.

Click **Remove bootstrap policy** to discontinue using secure conversation policy settings.

Configuring the request or response token policies:

You can configure the request and response token policies that are part of the WS-Security policy using the administrative console. Message requests token policies are applied to requests and enforced on responses to support both quality and interoperability.

Before you begin

You can configure some settings for the policies within your policy sets. The default policy sets provided in the product cannot be edited. You must create a copy of the default policy set or create a completely new policy set in order to specify the policies for it.

About this task

Use this administrative console task to define policies that specifically support security tokens and properties.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. Click **Services > Policy sets > Application policy sets > *policy_set_name* > WS-Security policy**.
2. Click one of the following links:
   - **Main policy** or
   - **Bootstrap policy**
   - Click the Main policy link to specify how message security policies are applied to requests and enforced on responses to support interoperability.
   - Click the Bootstrap policy link to configure how secure conversations are established. A bootstrap policy might already be configured. If no bootstrap policy is currently configured, first ensure that you have enabled message security with symmetric signature and encryption policies and secure conversation tokens for both integrity and confidentiality protection. See Configuring the WS-Security policy.
3. Click **Request token policies** under Request Policies or **Response token policies** under Response Policies. Use this to panel to define policies that specify which types of security tokens are supported for the properties of each token type.

Results

Once you have customized the WS-Security policy with the associated properties, including the request and response token policies, you can then send and receive protect messages.

Request or Response token policies collection:

Use this page to define policies that specify supporting security tokens and properties.

To view this administrative console page, complete the following:

1. Click **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name***.
2. Click the **WS-Security** policy in the policies table.
3. Click the **Main policy** link or the **Bootstrap policy** link.
4. Click **Request token policies** or **Response token policies** from the Policy Details section.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Add:

Add a token from a list of supported token types. You can only add a token to a custom policy set.

This option provides the following supported token types:

- Username
- X.509
- LTPA
- Custom

To change the settings for a token after adding it, click on the token identifer name in the table.

Delete:

Removes the selected token name. This action is only available for a custom policy set.

Token Identifier:

Specifies the name of the token.

Type:

Specifies the token type in the Supported token types table. This list displays a token type for each token name in the list.

Version:

Specifies the version of the token in the Supported token types table. This list displays the version of each token in the list.

Token type settings:

Use the administrative console to define the details about the token types. This panel is displayed differently for each different token type. Policies can be defined that specify which types of security tokens are supported as well as properties for the token type.

To view token types for a policy set, complete the following steps:

1. Click **Services > Policy sets > Application policy sets > *policy_set_name***.
2. Click the **WS-Security** policy in the Policies table.
3. Click the **Main policy** link or the **Bootstrap policy** link.
4. Click one of the following:
   - **Request token policies** from the Policy detail section.
   - **Response token policies** from the Policy detail section.
   - **Symmetric signature and encryption policies** from the Key symmetry section.
   - **Asymmetric signature and encryption policies** from the Key symmetry section.
5. For a Request token policy or a Response token policy, click a token from the Supported Token Types table or click the **Add Token Type** button to select the type of token to add.
6. For a symmetric signature and encryption policy or an asymmetric signature and encryption policy, click **Edit Selected Type Policy**.

This panel is displayed for each token type you are configuring or adding. It displays fields for some token types and not for others. This help topic contains all of the fields for each of the token types and describes which token is being configured for each field.

Custom token name:

For a custom token, specify the name of the token being configured. Enter or edit the name for the custom token in this entry field.

Local name:

For a custom token, specify the local name.

If the custom token type is used to generate a Kerberos token as defined in the OASIS Web Services Security Specification for Kerberos Token Profile v1.1, use one of the values in the following table for the local name. The value you choose depends on the specification level of the Kerberos token generated by the Key Distribution Center (KDC). The table lists the values and the specification level associated with each value. For purposes of interoperability, the Basic Security Profile V1.1 standard requires the use of the local name, http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ.

Local Name Value for Kerberos Token	Associated Specification Level
http://docs.oasis-open.org/wss/oasiswss-kerberos-token-profile-1.1#Kerb erosv5_AP_REQ	Kerberos V5 AP-REQ as defined in the Kerberos specification. This value is used when the Kerberos ticket is an AP Request.
http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ	GSS-API Kerberos V5 mechanism token containing a KRB_AP_REQ message as defined in RFC-1964 [1964], Sec. 1.1 and its successor RFC-4121, Sec. 4.1. This value is used when the Kerberos ticket is an AP Request (ST + Authenticator).
http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510	Kerberos V5 AP-REQ as defined in RFC1510. This value is used when the Kerberos ticket is an AP Request per RFC1510.
http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510	GSS-API Kerberos V5 mechanism token containing a KRB_AP_REQ message as defined in RFC-1964, Sec. 1.1 and its successor RFC-4121, Sec. 4.1. This value is used when the Kerberos ticket is an AP Request (ST + Authenticator) per RFC1510.
http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120	Kerberos V5 AP-REQ as defined in RFC4120. This value is used when the Kerberos ticket is an AP Request per RFC4120.
http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120	GSS-API Kerberos V5 mechanism token containing an KRB_AP_REQ message as defined in RFC-1964, Sec. 1.1 and its successor RFC-4121, Sec. 4.1. This value is used when the Kerberos ticket is an AP Request (ST + Authenticator) per RFC4120.

URI:

For a custom token, specify the uniform resource identifier (URI).

Leave this field empty, if the custom token type is used to generate a Kerberos token as defined in the OASIS Web Services Security Specification for Kerberos Token Profile v1.1.

LTPA token name:

For an LTPA token, specify the name of the token being configured. Enter or edit the name for the LTPA token in this entry field.

Propagate the JAAS subject:

For an LTPA token, specify whether the associated Java Authentication and Authorization Service (JAAS) subject is propagated. Select this check box to propagate the JAAS subject. The default value is not selected. Therefore, the JAAS subject is not propagated by default.

Username token name: Specify the name of the token being configured. Enter or edit the name for the username token in this entry field.

WS-Security version:

For a Username token, specify the version of Web Services Security, the WS-Security specification, that is used to secure the message transmission.

The following versions are available:

- WS-Security V1.0

- WS-Security V1.1

X.509 token name:

For a X.509 token, specify the name of the token being configured. Enter or edit the name for the X.509 token in this entry field.

WS-Security version:

For a X.509 token, specify the version of Web Services Security that is used to secure the message transmission.

The following versions are available:
- WS-Security V1.0
- WS-Security V1.1

X.509 type:

For a X.509 token, specify the type of X.509 token being configured.

The following types are available for the X.509 token:
- X.509 Version 1. This option is available with WS-Security Version 1.1 only.
- X.509, Version 3
- X.509 PKCX7
- PKI Path Version 1

Secure conversation token: The secure conversation token is available only when using symmetric signature and encryption policies.

Require reference to secure context token issuer:

For a secure conversation token, select this option to specify a reference to the issuer of the security context token.

After selecting the **Require reference to secure context token issuer** option, specify the URI of the security context token issuer.

Transform algorithms settings:

Use this administrative console page to select the uniform resource locator (URL) for the transform algorithms that are needed to protect the message part.

To view this administrative console page, complete the following:
1. Click **Services > Policy sets > General provider policy set bindings** or **General client policy set bindings**.
2. Click the name of the binding you want to edit.
3. Click **WS-Security** policy in the Policies table
4. Click the **Authentication and Protection** link in the Main message security policy bindings section.
5. Select a signature protection in the Request message and encryption protection section or the Response message signature and encryption protection section.
6. Click the **Additional bindings > Signed part reference default** at the end of the panel.
7. Click a URL in the Transform algorithms table.

You can also get to this administrative console page by completing the following actions:

1. Click **Applications > Application Types > WebSphere enterprise applications**.
2. Select an application that contains web services.
3. Select **Service provider policy sets and bindings** or **Service client policy sets and bindings**.
4. Select a binding.

   Important: You must have previously attached a policy set and assigned an application specific binding before you can do this step.
5. Select **WS-Security**.

   Important: You must have previously added WS-Security to the bindings.
6. Click the **Authentication and Protection** link in the Main message security policy bindings section.
7. Select a signature protection in the Request message and encryption protection section or the Response message signature and encryption protection section.
8. Click the **Additional bindings > Signed part reference default** at the bottom of the panel.
9. Click **New** to add a transform.
10. [Optional] Click a URL in the Transform algorithms table. This step assumes that a transform algorithm has previously been configured.

URL:

Specifies the URL for the transform algorithms that are used to protect the message part.

The URLs of the supported transform algorithms are listed in the table. The recommended transform is *http://www.w3.org/2001/10/xml-exc-c14n#* and it is the Exclusive XML Canonicalization, as well as the default value provided in the URL field when a transform is added.

List of URLs for the transform algorithms
http://www.w3.org/2001/10/xml-exc-c14n#
http://www.w3.org/TR/1999/REC-xpath-19991116
http://www.w3.org/2002/06/xmldsig-filter2
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform
http://www.w3.org/2002/07/decrypt#XML
http://www.w3.org/2000/09/xmldsig#enveloped-signature

Custom properties:

Specifies the custom properties name and value pair for the transform algorithms. Click **New** to add a new custom property. Click **Delete** to delete a custom property.

Name Specifies the name of a custom property that you choose to set to protect the message part.

Value Specifies the value of a custom property that you choose to set to protect the message part.

Signed part reference default bindings settings:

Use this administrative console page to configure the signed part reference general bindings and the uniform resource locator (URL) for the transform algorithms that are needed to protect the message part.

You can view and configure bindings for WS-Security using this administrative console page:

1. Click **Applications > Application Types > WebSphere enterprise applications**.
2. Select an application that contains web services

3. Select **Service provider policy sets and bindings** or **Service client policy sets and bindings**
4. Select a binding.

   Note: You must have previously attached a policy set and assigned an application specific binding before you can do this step.
5. Select **WS-Security**.

   Note: You must have previously added WS-Security to the bindings.
6. Click the **Authentication and Protection** link in the Main message security policy bindings section.
7. Select a signature protection in the Request message and encryption protection section or the Response message signature and encryption protection section.
8. Click the **Additional bindings - Signed part reference default** at the bottom of the panel. For the custom, select an assigned message part and click **Edit**. In the custom version, there is also a reference field, **Specifies the name of the signature protection in the policy set**. Use this field to specify the name of the signature protection in the policy set.
9. Check the **Include timestamp** check box to include the timestamp when the message part is signed.
10. Check the **Include nonce** check box to include the nonce when the message part is signed.

Part reference properties:

Specifies the name of the part reference properties that include the transform algorithms used to protect the message part.

1. Check the **Include timestamp** check box to include the timestamp for the message part.
2. Check the **Include nonce** check box to include the nonce when the message part is signed.

Transform algorithms:

Specifies the uniform resource locator (URL) description for the transform algorithms. Click the URL to view the transform algorithms.

Main policy and bootstrap policy settings:

Use this page to specify how message security policies are applied to requests and enforced on responses, as defined by the main policy settings and the bootstrap policy settings. Assertions for Web Services Security (WS-Security) versions are already generated based on assertions in the policy set. If the policy set includes a Web Services Security Version 1.1 assertion, then Web Services Security Version 1.1, itself, is asserted.

To view this administrative console page, use one of the following steps:

1. Click **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** .
2. Click the **WS-Security** policy in the Policies table.
3. Click the **Main policy** link or the **Bootstrap policy** link.

Message level protection:

Specifies whether message level protection, using digital signatures and encryption, is required.

Require signature confirmation
Specifies whether the signature confirmation is required. Select this check box to require a signature confirmation.

Message part protection:

Specifies whether message part protection, using digital signatures and encryption, is required.

Request message part protection
Click this link to define which request message parts you want protected and how that protection is provided.

Response message part protection
Click this link to define policies that specify which response message parts you want protected and how that protection is provided.

When the Message level protection check box is cleared, the link to Request message part protection is disabled, because the configuration information associated with message level security is removed when message level protection is cleared.

Key symmetry – Use symmetric tokens:

Specifies whether to use symmetric tokens. Select this radio button to use symmetric tokens. You can then configure symmetric tokens using the **Symmetric signature and encryption policies** link. Click this link to access the Symmetric signature and encryption policies panel where you can create the trust context in which to use symmetric tokens. Using the same token for signing and validating messages and encrypting and decrypting messages provides higher performance than can be achieved with asymmetric tokens. Use symmetric tokens within a trust context. If a custom Kerberos token type is used, you must select the Use symmetric tokens option.

Key symmetry – Use asymmetric tokens:

Specifies whether to use asymmetric tokens. Select this button to use asymmetric tokens. You can then configure asymmetric tokens using the **Asymmetric signature and encryption policies** link. Click this link to access the Asymmetric signature and encryption policies panel where you can create the trust context (message integrity and confidentiality) in which to use asymmetric tokens. Specify which token type to use for the initiator and recipient signature as well as the initiator and recipient encryption.

Include time stamp in security header:

Specifies whether to use a time stamp in the header. Select this check box to include a time stamp in the header. You can then specify where in the header to place the time stamp by using the **Security header layout** radio buttons.

Security header layout:

Specifies the layout rules for the security header.

You can use the following radio buttons for the security header layout:

Strict: declarations must precede use
The declarations in the header must precede the use.

Layout (Lax): order of contents can vary
The order of contents in the header can vary.

Lax but timestamp required first in header
The timestamp must be first in the header, but the order of the remaining elements can vary.

Lax but timestamp required last in header
The timestamp must be last in the header, but the order of the remaining elements can vary.

Policy details:

Specifies links for accessing the request token policies, response token policies, and algorithms for asymmetric tokens. Click these links to view token policies and cannonicalization algorithms that are supported. Algorithms are used to reconcile XML differences.

Request token policies:

Click this link to define policies that specify which types of supporting authentication tokens are used in the request and the properties of those token types.

Response token policies:

Click this link to define policies that specify which types of supporting authentication tokens are used in the response and the properties of those token types.

Algorithms for symmetric or asymmetric tokens:

Links to a view of available algorithms. Click this link to view the cryptographic and cannonicalization algorithms that are supported. Algorithms are used to reconcile XML differences.

Asymmetric signature and encryption policies settings:

Use this page to create the trust context, message integrity and confidentiality, to use asymmetric tokens. You can create the trust context by specifying which token type to use for the initiator and recipient signature as well as the initiator and recipient encryption.

To view this administrative console page complete the following actions:

1. Click **Services > Policy sets > Application policy sets > *policy_set_name* > WS-Security policy type**.
2. Click **Main policy** or **Bootstrap policy**.
3. Click the **Asymmetric signature and encryption policies** link.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Message integrity policies – Initiator signature token:

Specifies the token type of the initiator signature token. To add a token type or change the current token type that is displayed in the **Initiator signature token** field, or to edit the displayed token type, click **Action**.

Message integrity policies – Recipient signature token:

Specifies the token type of the recipient signature token. To add a token type or change the current token type that is displayed in the **Recipient signature token** field, or to edit the displayed token type, click **Action**.

Message confidentiality policies – Use the same token types that are used for integrity protection:

Specifies whether the token type set for initiator signature token and recipient signature token are used for the initiator encryption token and the recipient encryption token. When the box is checked the fields are empty and are cleared of data when **Ok** or **Apply** is selected.

Message confidentiality policies – Initiator encryption token:

Specifies the initiator encryption token type. To add a token type or change the current token type that is displayed in the **Initiator encryption token** field, or to edit the displayed token type, click **Action**.

Message confidentiality policies – Recipient encryption token:

Specifies recipient encryption token type. To add a token type or change the current token type that is displayed in the **Recipient encryption token** field, or to edit the displayed token type, click **Action**.

Action:

Specifies an option for each of the signature and encryption token fields. Use the **Action** button to change, delete, add, or edit the listed token type.

The Action button lists supported token types and provides the following options:

Edit selected type policy
Opens a page to edit the token type shown in the signature or encryption token fields.

Delete selected type policy
Removes the token type from the signature or encryption token fields.

Change to custom type
Opens the Custom type page to specify the uniform resource identifier (URI) for a custom token type.

Add custom type
Adds the custom type entry in the signature or encryption token fields.

Change to X.509
Changes the listed token type to X.509.

Add X.509
Adds the X.509 token type.

When you change the token type, any values you specified for the former token type are lost and the default values for the newly assigned token type are used.

Symmetric signature and encryption policies settings:

Use this page to create the trust context to use symmetric tokens. Using the same token for signing and validating messages and encrypting and decrypting messages increases performance. Use symmetric tokens within a trust context.

To view this administrative console page, complete the following options:
1. Click **Services > Policy sets > Application policy sets**.
2. Select a ***policy_set_name*** in the policy sets table that contains WS-Security content.
3. Click **WS-Security** in the policies table.
4. Click the **Main policy** link or the **Bootstrap policy** link.
5. Click the **Symmetric signature and encryption policies** link.

Message Integrity – Token type for signing and validating messages:

Specifies the current token type used for signing and validating messages.

To change the current token type that is displayed in the **Token type for signing and validating messages** field or to edit the displayed token type, click **Action**.

Message Confidentiality – Use same token type for confidentiality that is used for integrity:

Specifies whether the token type set for signing and validating messages is also used for encrypting and decrypting messages. For a Kerberos token, message confidentiality uses the same token that is used for the message integrity.

If you select this check box, then the **Token type for encrypting and decrypting messages** field is blank. If you clear this check box, then a different token can be used for message confidentiality.

Message Confidentiality – Token type for encrypting and decrypting messages:

Specifies the current token type that is used for encrypting and decrypting messages.

To change the current token type that is displayed in the **Token type for encrypting and decrypting messages** field or to edit the displayed token type, verify that the **Use the same token type for confidentiality that is used for integrity protection** check box is cleared, and click **Action**.

Action:

Enables the token type selected to be changed or edited.

The **Action** button lists supported token types and provides the following options:

Edit selected type policy
Opens a page to edit the token type for signing or encrypting fields.

Change to Secure Conversation
Changes the token type to Secure Conversation.

Change to custom type
Opens the custom type page to specify the Uniform Resource Identifier (URI) for a custom token type.

When you change the token type, any values that you specified for the former token type are lost, and the default values for the newly assigned token type are used.

Algorithms settings:

Use this page to view the supported cryptographic and cannonicalization algorithms. Algorithms are used to reconcile XML differences.

To view this administrative console page:
1. Click **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name***.
2. Click the **WS-Security** policy in the Policies table.
3. Click the **Main policy** link or the **Bootstrap policy** link.
4. Click the **Algorithms for symmetric tokens** link or the **Algorithms for asymmetric tokens** link.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

Algorithm suite:

Specifies the supported algorithms that are required for performing cryptographic operations with symmetric or asymmetric key-based security tokens.

All of the algorithm values in this field specify an algorithm suite. Algorithm suites and the values they each represent are detailed in the *Web Services Security Policy Language (WS-SecurityPolicy) July 2005 Version 1.1* specification. Select a supported algorithm from the following list:
- Basic256
- Basic192
- Basic128
- TripleDes
- Basic256Rsa15
- Basic192Rsa15
- Basic128Rsa15

- TripleDesRsa15
- Basic256Sha256
- Basic192Sha256
- Basic128Sha256
- TripleDesSha256
- Basic256Sha256Rsa15
- Basic192Sha256Rsa15
- Basic128Sha256Rsa15
- TripleDesSha256Rsa15

This table defines values for the components for each algorithm suite.

Table 258. Algorithm suite components. The algorithms are used to perform cryptographic operations on tokens.

Algorithm Suite	Digest	Encryption	Symmetric Key Wrap	Asymmetric Key Wrap	Encryption key Derivation	Signature key Derivation	Minimum Symmetric Key Length
Basic256	Sha1	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192	Sha1	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128	Sha1	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDes	Sha1	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Rsa15	Sha1	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Rsa15	Sha1	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Rsa15	Sha1	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesRsa15	Sha1	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192
Basic256Sha256	Sha256	Aes256	KwAes256	KwRsaOaep	PSha1L256	PSha1L192	256
Basic192Sha256	Sha256	Aes192	KwAes192	KwRsaOaep	PSha1L192	PSha1L192	192
Basic128Sha256	Sha256	Aes128	KwAes128	KwRsaOaep	PSha1L128	PSha1L128	128
TripleDesSha256	Sha256	TripleDes	KwTripleDes	KwRsaOaep	PSha1L192	PSha1L192	192
Basic256Sha256Rsa15	Sha256	Aes256	KwAes256	KwRsa15	PSha1L256	PSha1L192	256
Basic192Sha256Rsa15	Sha256	Aes192	KwAes192	KwRsa15	PSha1L192	PSha1L192	192
Basic128Sha256Rsa15	Sha256	Aes128	KwAes128	KwRsa15	PSha1L128	PSha1L128	128
TripleDesSha256Rsa15	Sha256	TripleDes	KwTripleDes	KwRsa15	PSha1L192	PSha1L192	192

When using a Kerberos custom token based on the OASIS Web Services Security Specification for Kerberos Token Profile V1.1, only Aes128, Aes256, and TripleDes encryption-based algorithm suites are supported.

Cannonicalization algorithm:

Specifies whether to use inclusive or exclusive cannonicalization.

The following supported cannonicalization algorithms are available in this list:

- Exclusive cannonicalization
- Inclusive cannonicalization

The default value is `Exclusive cannonicalization`.

XPath version:

Specifies the version of the XPath filter to use.

The following supported XPath versions are available:

- XPath 1.0
- XPathfilter 2.0

The `XPathfilter 2.0` version is the default value.

Use security token reference transformation:

Specifies whether the security token reference is transformed. Indicate whether the security token reference transform is either `True` or `False`.

Message part protection settings:

Use this page to define the message parts that you want protected and how that protection is provided.

To view this administrative console page, complete the following actions:
1. Click **Services > Policy sets > Application policy sets** >*policy_set_name*.
2. Click the **WS-Security** policy in the Policies table.
3. Click the **Main policy** link or the **Bootstrap policy** link.
4. Click the **Request message part protection** link or the **Response message part protection** link from the Message Part Protection section.

Integrity protection – Signed parts:

Specifies the signed parts that you have added for message integrity protection. You can select parts to edit or delete, or add new signed parts.

Button	Resulting Action
Add	Specifies each element of a new signed part and to add those elements to the Signed parts listing.
Edit	Loads the specific part you selected for editing.
Delete	Removes the selected signed part from the list.

Confidentiality protection – Encrypted parts:

Specifies the encrypted parts that have been added for message confidentiality protection. You can select parts to edit or delete, or add new encrypted parts.

Button	Resulting Action
Add	Enables you to specify each of the elements of a new encrypted part and to add those elements to the Encrypted parts listing.
Edit	Loads the specific part you selected for editing.
Delete	Removes the selected encrypted part from the list.

Signed part settings:

Use this page to define the elements of a signed part. Signed parts are used to protect message integrity and, in this case, the signed parts are being defined as part of the policy set process.

To view this administrative console page, complete the following actions:
1. Click **Services > Policy sets > Application policy sets > *policy_set_name***.
2. Click the **WS-Security** policy in the Policies table.
3. Click either the **Main policy** link or the **Bootstrap policy** link.
4. Click the **Request message part protection** link or the **Response message part protection** link in the Message Part Protection section.
5. In the Integrity protection section, complete one of the following actions:

- Click **Add** to add a new signed part.
- Select an existing signed part and click **Edit**.

Name of part to be signed:

Specifies the name of this set of one of more message parts that you have selected to sign. The name you choose is a label and must be unique within the Response message part protection or Request message part protection collections for this WS-Security policy.

Elements in part:

Specifies a list of the message elements included in the signed part. The **Elements in part** field contains a listing of message elements that are included in this signed part to provide message integrity.

Click **Add** to add an element to the signed part of the message. To remove a message element from a signed part of a message, first click the selection box next to the element to be removed, then click **Remove**. Use the **OK**, **Apply**, **Reset** or **Cancel** buttons for the text entry fields. The QName or the Xpath expression value is required and can be edited at any time, such as when adding a new element, or after the element is added.

Protect message body:

Specifies if the message body is protected in this part. To protect the message body in this part, click **Protect message body**.

XPath expression:

Specifies if the displayed XPath expression is used as the method for specifying that a specific element is included in this part.

Select **XPath** from the **Add** menu list and provide an expression in the new XPath entry that is displayed in the table. Any Xpath expression row on the table that has no corresponding value is removed when you click **OK** or **Apply**.

QName for SOAP header elements only:

Specifies the Qname type for a namespace value for the SOAP header element that you want to encrypt. To encrypt a SOAP header element, select **Qname** and provide the **namespace** and optionally the **localname** of the SOAP header element in the **Value** field. When specifying the Qname, if using the optional localname, a comma must be inserted between the namespace and the localname, for example **<namespace>,<localname>**. If the **localname** is omitted, all SOAP header elements with the specified **namespace** are encrypted. To use the **Qname** selection method, the SOAP header elements must be the immediate children of the SOAP header. Any Qname row in the table that has no corresponding value is removed when you click **OK** or **Apply**.

Restriction:
You cannot select header elements that are sub-elements of other elements in the SOAP header using **Qname**. In this case, you must use an **Xpath expression** to select these header elements.

Encrypted message part settings:

Use this page to define the elements of an encrypted part of a message. Encrypted parts are used to protect message confidentiality, and in this case, the encrypted parts are being defined as part of the policy set process. A message part is a named set of one or more message elements.

To view this administrative console page, complete the following actions:

1. Click **Services > Policy sets > Application policy sets > *policy_set_name***.
2. Click the **WS-Security** policy in the Policies table.
3. Click the **Main policy** link or the **Bootstrap policy** link.
4. Click the **Request message part protection** link or the **Response message part protection** link in the Message Part Protection section.
5. In the Confidentiality protection section, you can perform any of the following:
   - Click **Add** to add a new encrypted part.
   - Select an existing encrypted part, and click **Edit**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Name of part to encrypt:

Specifies the name of the set of one or more message parts that you have selected to encrypt. The name you choose is a label and must be unique within the Response message part protection or Request message part protection collections for this WS-Security policy.

Elements in part:

Specifies a list of the message elements that are included in the encrypted part. The **Elements in part** field contains a listing of message elements that are included in this encrypted part to provide message confidentiality.

Click **Add** to add an element to the encrypted part of the message. To remove a message element from an encrypted part of a message, first click the selection box next to the element to be removed, then click **Remove**. The value of the Qname namespace, or the Xpath expression, is required and can be edited at any time, while adding a new element or after the element is added.

Body Specifies the body of the message part.

Qname for SOAP header elements only
Specifies the Qname type for a namespace value for the SOAP header element that you want to encrypt. To encrypt a SOAP header element, select **Qname** and provide the **namespace** and optionally the **localname** of the SOAP header element in the **Value** field. When specifying the Qname, if using the optional localname, a comma must be inserted between the namespace and the localname, for example **<namespace>,<localname>**. If the **localname** is omitted, all SOAP header elements with the specified **namespace** are encrypted. To use the **Qname** selection method, the SOAP header elements must be the immediate children of the SOAP header. Any Qname row in the table that has no corresponding value is removed when you click **OK** or **Apply**.

Restriction: You cannot select header elements that are sub-elements of other elements in the SOAP header using Qname. In this case, you must use an Xpath expression to select these header elements.

Xpath expression
Specifies if the displayed Xpath expression is used as the method for specifying that a specific element is included in this part. Select **XPath** from the **Add** menu list, and provide an expression in the new XPath entry that is displayed in the table. Any Xpath expression row on the table that has no corresponding value is removed when you click **OK** or **Apply**.

Configuring the Custom properties policy:

When working with policy sets in the administrative console, you can customize policies to set generic properties that are not supported in other policy types. You must customize the Custom properties policy configuration because all properties within the policy are user-defined.

About this task

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

You can configure generic properties that are not supported in other policy types with the Custom properties policy. These additional properties are set in the binding. The Custom properties policy provides an alternative way to set a binding property instead of using the JAX-WS programming model to set the property on the BindingProvider object. The CustomProperties binding is only supported for service clients.

After you have created a new policy set with the Custom properties policy added, you must create a Custom properties binding and add at least one custom property to configure the binding. A Custom properties binding type can be added to a general client binding or an application specific client binding. You cannot create an empty Custom properties binding. You must define at least one name and value pair within the binding.

Note: In a mixed cell environment, the following limitations apply to attachments to policy sets containing CustomProperties policy:

- You must not create attachments to policy sets containing CustomProperties policy for applications that are deployed on an application server that is prior to WebSphere Application Server Version 8. The CustomProperties policy is only supported on WebSphere Application Server V8 and later.
- An application that contains an attachment to a policy set containing CustomProperties policy must not be deployed on an application server that is prior to WebSphere Application Server Version 8.
- If an application that is deployed in a cluster environment contains an attachment to a policy set containing CustomProperties policy, you must not add a member application server that is prior to WebSphere Application Server Version 8 to the cluster.

Procedure

1. Create the Custom properties policy.
   a. Navigate to the Application policy sets or the System policy sets From the administrative console, click **Services** > **Policy Sets** > **Application policy sets** or **Services** > **Policy Sets** > **System policy sets**.
   b. Click the **New...** button.
   c. Enter a unique policy set name.
   d. Click the **Add** button and choose the **Custom properties** policy in the Policies table.
   e. Click the **Apply** button.
   f. Click the **Save** link.

   You cannot configure the Custom properties policy from this panel. You must configure your Custom properties with a Custom properties binding.
2. Customize the Custom properties bindings.
   a. Navigate to the Custom properties bindings. From the administrative console, click **Services** > **Policy Sets** > **General client policy set bindings** > ***client_policy_set_binding_name***.
   b. Click **Add** > **Custom properties**.

      The Custom properties (bindings) window displays options for adding Custom properties bindings.
   c. Specify the properties for each custom property you want to add. The following fields are required:

      Name Displays the name of the custom property.

Value Displays the value of the custom property.

The application specific bindings can be customized similarly, from the application policy set and bindings attachments panel.

Results

After you have customized the Custom properties policy, the associated policy set uses this policy to set generic properties that are not supported in other policy types.

Policy set bindings settings for Custom properties:

Use this page to view, define, or configure general bindings or application specific bindings, for the Custom properties policy.

This administrative console page applies only to Java API for XML Web Services (JAX-WS) applications.

To view this administrative console panel, click **Services** > **Policy sets** > **General client policy set bindings**.

To edit or configure the general bindings, complete the following steps:

1. Click **New**.
2. For **Bindings configuration name**, specify a unique binding name.
3. Optionally, specify a description.
4. Click **Add** and select the **Custom properties** policy in the Policies table.
5. Specify a unique name and value pair.
6. Optionally, to add additional properties, click **New**.
7. Optionally, to edit a property, select a property and click **Edit**.
8. Optionally, to delete a property, select a property and click **Delete**.
9. Click **Apply**.
10. Click **Save**.

Note: This product supports using the Custom properties policy and binding to set generic properties that are not supported in other policy types. The additional properties are set in the binding. The Custom properties policy provides an alternative way to set a binding property instead of using the JAX-WS programming model to set the property on the BindingProvider object. You cannot create an empty Custom properties binding. You must define at least one name and value pair within the binding. Additionally, there are no predefined properties in the binding because all properties are user-defined. The CustomProperties binding is only supported for service clients.

Note: In a mixed cell environment, the following limitations apply to attachments to policy sets containing CustomProperties policy:

- You must not create attachments to policy sets containing CustomProperties policy for applications that are deployed on an application server that is prior to WebSphere Application Server Version 8. The CustomProperties policy is only supported on WebSphere Application Server V8 and later.
- An application that contains an attachment to a policy set containing CustomProperties policy must not be deployed on an application server that is prior to WebSphere Application Server Version 8.
- If an application that is deployed in a cluster environment contains an attachment to a policy set containing CustomProperties policy, you must not add a member application server that is prior to WebSphere Application Server Version 8 to the cluster.

Name:

Specifies the name associated with this property.

Data type String

Value:

Specifies the value associated with this property.

Data type String

Enabling policies for policy sets using the administrative console

Policies can be listed in a policy set in the disabled state so that they are not currently included in the policy set. You can enable a policy to be included in a policy set using the administrative console.

Before you begin

To enable a policy for a policy set, be sure the policy is listed in the policy set and shown in the **Disabled** state in the **State** column of the **Policies** table on the Policy set settings page.

About this task

To enable a policy in a policy set, use the administrative console.

Procedure

1. Click **Services > Policy sets > Application policy sets** > ***policy_set_name*** or **Services > Policy sets > System policy sets** > ***policy_set_name***. The Policy Set Settings page displays a listing of available policies in the **Policies** table for the policy set selected. If this table contains no policies to enable, no policies exist for the policy set of interest. In this case, you must add the policies to the policy set.
2. Click the **Select** box beside the disabled policy that you want to enable. You can select multiple policies if you want to enable more than one.
3. Click the **Enable** button. The **State** column of the **Policies** table is updated to display the selected policy as enabled.

Results

You have enabled a policy for the selected policy set.

What to do next

If the policy is not listed in the Policies table, it cannot be enabled and must be added. You can modify the policy after it is enabled.

Disabling policies from policy sets using the administrative console

You can have policies listed in a policy set that are in the enabled state so that they are currently included in the policy set. You can disable a policy from being included in a policy set without deleting it from the policy set. You might want to do this if you want the policy included in the policy set in the future. You can use the administrative console to change this setting.

Before you begin

To disable a policy for a policy set, be sure the policy is listed in the policy set and shown in the **Enabled** state in the **State** column of the **Policies** table on the Policy set settings page. If the policy is not listed, it might have been deleted and you must add it again and then disable it.

About this task

You can disable a policy in a policy set, from the menu of the administrative console.

Procedure

1. Click **Services > Policy sets > Application policy sets** >***policy_set_name*** or **Services > Policy sets > System policy sets** >***policy_set_name***. Clicking a policy set name from the listing in the policy sets collection opens the Policy set settings panel for that policy set. This panel displays a listing of available and enabled policies in the **Policies** table for the policy set you selected. If this table contains no policies, there are no existing policies to disable.
2. Click the**Select** box beside the enabled policy to be disabled. You can select multiple policies if you want to disable more than one.
3. Click **Disable**. The **State** column of the **Policies** table is updated to display the selected policy as disabled. The policy was not deleted from the policy set and is still available to be enabled.

Results

You have disabled a policy for the selected policy set.

What to do next

You could now enable other policies or add or modify existing policies for this policy set.

Web services policies

Policies define the type of web service policy based on the quality of service type. Policies are initially set with default settings but the attributes can be edited and changed.

Provided policies include:

WS-Addressing
: Based on the World Wide Web Consortium (W3C) WS-Addressing specifications for web services. This family of specifications provide transport-neutral mechanisms to address web services and to facilitate end-to-end addressing. This specification provides asynchronous support.

WS-Security
: Based on the WS-Secure Conversation (WS-SC) and WS-Security specifications along with the associated token profiles. The WS-Security specification and its associated token profiles define a way to send security tokens and provide message integrity and confidentiality. The WS-Secure Conversation specification establishes a secure context, based on shared keys, for the client and server to use for a series of messages. This standard provides a framework across organizations that defines how to secure the entire conversation. Use the WS-Security policy to define how the SOAP messages are secured. It has options such as:

  - which message parts are signed and encrypted
  - the tokens types to be included
  - whether to use symmetric or asymmetric cryptography

  You can also use WS-Security policies to define the bootstrap policy that is used to acquire security context tokens. Security context tokens are used by secure conversation.

WS-Reliable Messaging (WS-RM)
: This specification enables the sender and receiver to assure the quality of services in a set of messages. It helps the application developer deal with latency issues, maintenance interruption, and other problems that prevent messages from being completed. This quality assurance is critical for stateful applications.

WS-Transaction
: This specification provides support for coordination of atomic transactions or business activities for

web services applications. You can enable WS-Transaction on both the client (outbound) and server (inbound) side by attaching a policy set that enables the WS-Transaction policy as part of the policy set. This policy is based on the WS-AtomicTransaction specification and the WS-BusinessActivity specification, together with the WS-Coordination specification.

HTTP Transport
The HTTP transport policy applies the HTTP features and HTTP connections polices to outbound messages. The response listener policy is enforced on inbound messages.

SSL Transport
Provides SSL transport security for the HTTP protocol with web services applications.

JMS Transport
When using the SOAP over JMS transport with JAX-WS applications, you can customize the transport by configuring the JMS transport policy. The SOAP over JMS transport provides an alternative to HTTPS for transporting SOAP requests and response messages between clients and servers. Use the JMS transport policy to configure a service that uses the JMS transport to send asynchronous response messages back to the client.

Custom Properties
Provides the ability to specify generic binding properties for web service applications. The CustomProperties binding is only supported for service clients.

Exporting policy sets using the administrative console

You can export policy sets between a client and a provider or between servers using the administrative console.

Before you begin

Before you begin this task, select the policy set to be exported. Read about the application policy sets collection and exporting policy sets to client or server environments using the wsadmin tool.

About this task

Static export is used in development environment to exchange a policy sets between a client and a provider. You can also export between servers. The exported format is a .zip file.

Procedure

1. From the administrative console navigation, click **Services** > **Policy sets** > **Application policy sets** or **Services** > **Policy sets** > **System policy sets**. The Application policy sets collection page displays a listing of the custom and default policy sets. Custom policy sets are displayed only if you have created them. If you have not created a custom policy set, then only the default policy sets are displayed.
2. From the Application policy sets panel, select a policy set and click **Export**.
3. Locate the export archive file. From the Export policy set archive file settings panel, click the archive file to export.
4. Choose a location to export the file. Do not export the file to a browser.

Results

When you finish this task, the policy set archive file has been exported to the location you specified.

Example

If you have a policy set, `ABC_ps` and you want to move it from `ServerA` to `ClientA`, you can use the export function. The export policy set can be used by importing it into another application server. Read about the command task for importing an exported policy set compression file.

What to do next

You can import the policy set to reuse it.

Application policy sets collection

Use this page to manage policy sets. You can create, copy, export, and import policy sets. You can also view or delete existing policy sets. You can use policy sets, or assertions that define services, to simplify your web services configuration because policy sets group security and other web services settings into reusable units.

To view this administrative console page, click **Services** > **Policy sets** > **Application policy sets**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Name

Specifies a list of available policy sets. Click a policy set name to access the details panel for that policy set. Use the buttons on the page to create a new policy set or manage the existing policy sets. You can also import a policy set to display in the list of available policy sets.

The following list of buttons are available to you for creating and managing policy sets:

Button	Resulting action
New	Creates a new policy set. This option opens an empty policy set and allows you to provide additional details, such as the name, description, and policy configuration options.
Delete	Removes the selected policy set.
Copy	Creates a copy of the policy set that is selected from a list. After you create a copy, you can provide a name and a description for the copy. You must also specify whether to transfer attachments and binding information, where these options are available, from the original to the copy.
Import	Imports a policy set. This is a menu item with the option of importing a policy set from a default repository or a selected location. You can select and import the default policy sets from the default repository. The default repository for the import function in the administrative console is the directory that contains the default policy sets. Initially, a subset of pre-configured policy sets are exposed. However, there are other default policy sets that are available to import when you select **Import** > **From Default Repository**. The administrative console also displays the default policy sets in a list that includes descriptions, that you can use to select the policy set that you want to import.
Export	Exports the selected policy set to an archive file.

Editable

This column shows whether the policy set is a user-defined, custom policy set that can be edited or whether the policy set is a default policy set that is not editable.

Values displayed in this field are `Editable` or `Not editable`. Even though a policy set is identified as not editable, it is deletable. For example, you cannot edit information for the default system policy set, but you can delete the policy set. You can change the properties for a default, not editable policy set by copying it, and then modifying the properties of the copy. For more information on modifying a default policy set, see copy of default policy set and bindings settings document.

Data type: String

Default: Not editable

Description
Provides a brief description of the application policy sets that currently exist. You cannot edit information for the default application policy sets; however, you can delete the policy sets. The list of contained policies for each policy set that it relates to are displayed in the Name column. The contained policies that are displayed are only those that are enabled. From this panel, the field is not editable.

For custom policy sets that you create or modify, you can edit this description when you create or customize policy sets on the details panel. The description is the same one that you entered in the **Description** field when you created that policy set. The included and enabled policies appear at the top followed by a bulleted list of the key attributes of the policy set.

Application policy set settings
Use this page to view, create, enable or disable your policy sets. You can use policies, or assertions that define services, to simplify your web services configuration.

To view this administrative console page, click one of the following paths:
- **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name***.
- **Services** > **Policy sets** > **Application policy sets** > **New**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Important: You can only edit the fields on this page if you open this page for a custom policy set. You cannot edit default policies.

Policy set name
Specifies the name of a selected policy set. This field is empty if you are creating a new policy set and you can enter a policy set name. If you select an existing policy set from the Policy set page, then this field displays the name of that policy set. This field is also not editable for existing custom policy sets and default policy sets.

Description
Specifies a brief description of the policy set being viewed, modified or created. This field displays a brief description of the policy set displayed in the Policy set name field or is empty if a new policy set is being created. You can edit this field for custom policy sets only.

Policies
The Policies table displays a list of policies that are contained in the policy set. This table initially contains no policies when you are creating a new custom policy set. If this is a default policy set, then you cannot add, delete, enable or disable the policy. The configuration of a policy in a default policy set cannot be altered but you can view a policy by clicking on the policy name link. If the policy is a custom policy, you can modify the configuration of the policy by clicking the policy name link. You can alter the policies contained in a custom policy set using the following buttons:

Button	Resulting Action
Add	Provides a list of valid policies, instances of which are not already included in the collection, that can be added to the policy set collection. The following policies are available: • Custom properties • HTTP transport • JMS transport • SSL transport • WS-Addressing • WS-ReliableMessaging • WS-Security • WS-Transaction
Delete	Removes the selected policy from the policy set and from the listing in the Description column of the table.
Enable	The policy is enabled for the policy set and displayed as enabled in the State column.
Disable	The policy is disabled for the policy set. The policy remains in the list but the State field shows the policy as Disabled.

Policies - Policy
Specifies the name of a policy. This field is empty if you are creating a new policy set. If you select an existing policy, then this field displays the name of that policy. If the policy settings are viewable, then you can click on the policy name to view the settings. For custom policy sets, you can click on the policy name to configure the policy settings.

Policies - State
Specifies an Enabled or Disabled status for each policy in the policy set. You cannot change the state for default policy sets. To change the state of the policy, select the policy and click **Enable** or **Disable**. The state is refreshed in this column to reflect your change.

If you disable a policy, you are temporarily removing the policy from the policy set without losing the configuration details for that policy. When a policy is disabled, you can easily enable the policy again by selecting **Enable**. If the policy is deleted, the configuration for that policy no longer exists. After you delete a policy, to enable that policy in a policy set, you must add the policy to the policy set and enter the configuration details for that policy.

Policies - Description
Specifies a brief description of each policy in the policies table. The product does not support custom policies; therefore, this field is not editable. You cannot create a new policy.

Additional properties – Attached deployed assets
Click this link to search for applications to which this policy set is attached. You can also find applications that contain service resources to which the policy set is attached.

Search attached applications collection
Use this page to search for applications and other resources that are attached to a specific policy set or to search for applications and other resources that have attached service resources.

To view this administrative console page, complete the following actions:

1. Click **Services > Policy sets > Application policy sets** >*policy_set_name*.
2. Click **Attached applications** link in the Additional Properties section.

Name
Specifies a list of applications that match the search text. The application names that are displayed in the Name column are either attached explicitly to the specified policy set or have service resources attached to this policy set.

To alter the policy set that is attached to an application, select an application, and click a button to enable the following options:

Button	Resulting Action
Detach Policy Set	Detaches the current policy set from the selected application or applications. This action also detaches any attached application service resources. This action does not detach other policy sets from the application or application service resources.
Replace Policy Set	Displays a list of policy sets that can be attached to the selected application or applications and any contained attached service resources. The policy set is replaced with the one selected. This action does not replace other policy sets that are attached to resources in the application that you select.

Click the application name to complete actions such as attaching policy sets to the application, its services, or its endpoints.

Web services policy sets
Policy sets are assertions about how services are defined. They are used to simplify your quality of service configuration for web services.

Note: You can use policy sets only with Java API for XML-Based Web Services (JAX-WS) applications. You cannot use policy sets with Java API for XML-based RPC (JAX-RPC) applications.

Policy sets combine configuration settings, including those for transport and message level configuration, such as WS-Addressing, WS-ReliableMessaging, and WS-Security.

There are two main types of policy sets; application policy sets and system policy sets. Application policy sets are used for business-related assertions. These assertions are related to the business operations that are defined in the Web Services Description Language (WSDL) file. System policy sets, on the other hand, are used for non-business-related system messages. These messages are not related to the business operations that are defined in the WSDL, but instead refer to messages that are defined in other specifications which apply qualities of service (QoS). Such QoS are the request security token (RST) messages that are defined in WS-Trust, or create sequence messages that are defined in WS-Reliable Messaging metadata exchange messages of the WS-MetadataExchange.

Policies are defined based on a quality of service. Policy definition is typically based on WS-Policy standard language, for example, the WS-Security policy is based on the current WS-SecurityPolicy from the Organization for the Advancement of Structured Information Standards (OASIS) standards.

An instance of a policy set consists of a collection of policies. For example, the WS-I RSP default policy set consists of instances of the WS-Security, WS-Addressing, and WS-ReliableMessaging policy types. A policy set is identified by a unique name that is unique across the cell. An empty policy set is a policy set with no policies defined.

You can use a default policy set after it is imported. If you want to change the properties for a default, not editable policy set, you need to copy the policy set to create an editable version to modify. See copy of default policy set and bindings settings. You can perform the following actions on policy sets:

- create
- copy
- edit
- delete
- attach to service resources like applications
- detach from service resources like applications
- export
- import

Note which functions you can configure using policy sets and the relationship of the security information that is configured. A set of default policy sets are included that you can import; then copy and rename for reuse. You can use a default policy set after it is imported, but if you want to change any of the settings, you need to copy the policy set to create an editable version. The configuration can then be altered and customized on the copy.

Important: You can only copy and customize policy sets using the administrative console or administrative commands. Policy sets do not function correctly if they are copied manually.

On the application server, policy sets are stored at the cell level. Policy sets are centrally located so that they are available to all applications on the server.

The following application policy sets are installed on the base or network deployment (ND) profile by default: WS-I RSP or WS-I RSP (ND), Username WSSecurity default, and WSHTTPS default. The WS-I RSP (ND) is installed in a network deployment environment.

The following policy sets are ready for you to use as is.
- LTPA WSSecurity Default
- Kerberos V5 HTTPS default
- SSL WSTransaction
- Username SecureConversation
- Username WSSecurity default
- WS-Addressing default
- WSHTTPS default
- WS-I RSP ND
- WS-ReliableMessaging persistent

The application server also provides other default policy sets that you can use or customize. To use the additional policy sets, you must import them from the default repository. Read about importing policy sets from the administrative console for more information.

The following default policy sets are provided:

WS-I RSP default

This policy set provides:
- Reliable message delivery to the intended receiver by enabling WS-ReliableMessaging
- Message integrity through digital signature that includes signing the body, time stamp, WS-Addressing headers and WS-ReliableMessaging headers using the WS-SecureConversation and WS-Security specifications
- Confidentiality through encryption that includes encrypting the body, signature elements, using the WS-SecureConversation and WS-Security specifications

LTPA WS-I RSP default

This policy set provides:

- Reliable message delivery to the intended receiver by enabling WS-ReliableMessaging
- Message integrity through digital signature that includes signing the body, time stamp, WS-Addressing headers and WS-ReliableMessaging headers using the WS-SecureConversation and WS-Security specifications
- Confidentiality through encryption that includes encrypting the body, signature elements, using the WS-SecureConversation and WS-Security specifications
- A Lightweight Third Party Authentication (LTPA) token included in the request message to authenticate the client to the service

Username WS-I RSP default

This policy set provides:

- Reliable message delivery to the intended receiver by enabling WS-ReliableMessaging
- Message integrity through digital signature that includes signing the body, time stamp, WS-Addressing headers and WS-ReliableMessaging headers using the WS-SecureConversation and WS-Security specifications
- Confidentiality through encryption that includes encrypting the body, signature elements, using the WS-SecureConversation and WS-Security specifications
- A username token included in the request message to authenticate the client to the service. The username token is encrypted in the request

SecureConversation

This policy set provides:

- Message integrity through digital signature that includes signing the body, time stamp, and WS-Addressing headers using WS-SecureConversation and WS-Security specifications
- Message confidentiality through encryption that includes encrypting the body, signature and signature confirmation elements, using WS-SecureConversation and WS-Security specifications

LTPA SecureConversation

This policy set provides:

- Message integrity through digital signature that includes signing the body, time stamp, and WS-Addressing headers using WS-SecureConversation and WS-Security specifications
- Message confidentiality through encryption that includes encrypting the body, signature and signature confirmation elements, using WS-SecureConversation and WS-Security specifications
- A Lightweight Third Party Authentication (LTPA) token included in the request message to authenticate the client to the service

Username SecureConversation

This policy set provides:

- Message integrity through digital signature that includes signing the body, time stamp, and WS-Addressing headers using WS-SecureConversation and WS-Security specifications
- Message confidentiality through encryption that includes encrypting the body, signature and signature confirmation elements, using WS-SecureConversation and WS-Security specifications
- A username token included in the request message to authenticate the client to the service. The username token is encrypted in the request

WSAddressing default

Enables WS-Addressing support, which uses endpoint references and message addressing properties to facilitate the addressing of web services in a standard and interoperable way.

WSHTTPS default

Provides SSL transport security for the HTTP protocol with Web services applications.

Kerberos V5 HTTPS default

This policy set provides message authentication with a Kerberos Version 5 token. Message integrity and confidentiality are provided by Secure Sockets Layer (SSL) transport security. This policy set follows the OASIS Kerberos Token Profile V1.1 and WS-Security specifications.

When you use this policy set, configure the basic authentication data and custom properties such as the com.ibm.wsspi.wssecurity.krbtoken.targetServiceName and com.ibm.wsspi.wssecurity.krbtoken.targetServiceHost custom properties in the client bindings. For more information, see the Authentication generator or consumer token settings and Protection token settings (generator or consumer) topics.

Kerberos V5 SecureConversation

This policy set provides message integrity by digitally signing the body, time stamp, and WS-Addressing headers. Message confidentiality is provided by encrypting the body and the signature. The bootstrap policy is configured with the Kerberos V5 token. This policy set follows the WS-SecureConversation, OASIS specification for the Kerberos Token Profile, in addition to the WS-Security specification.

To use this policy set, you must also use the Client sample V2 and Provider sample V2 general sample bindings for your applications. For more information, refer to the topic General sample bindings for JAX-WS applications. To use this new policy set, create a new profile after installing the product.

To update existing profiles with this new policy set and the general bindings, Client sample V2 and Provider sample V2 general sample bindings, you must complete some manual steps. You only need to update the deployment manager profile and stand-alone application server profiles. To complete the manual steps for an existing profile, refer to the topic Configuring Kerberos policy sets and V2 general sample bindings.

Kerberos V5 WSSecurity default

This policy set provides message integrity by digitally signing the body, time stamp, and WS-Addressing headers. Message confidentiality is provided by encrypting the body and the signature using Advanced Encryption Standard (AES) encryption. The derived key from the Kerberos V5 token is used. This policy set follows the OASIS specification for the Kerberos Token Profile, in addition to the WS-Security specification.

To use this policy set, you must also use the Client sample V2 and Provider sample V2 general sample bindings for your applications. For more information, refer to the topic General sample bindings for JAX-WS applications. To use this new policy set, create a new profile after installing the product.

To update existing profiles with this new policy set and the general bindings, Client sample V2 and Provider sample V2 general sample bindings, you must complete some manual steps. You only need to update the deployment manager profile and stand-alone application server profiles. To complete the manual steps for an existing profile, refer to the topic Configuring Kerberos policy sets and V2 general sample bindings.

TrustServiceKerberosDefault

This policy set specifies the symmetric algorithm and the derived keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using the HMAC-SHA1 algorithm. Message confidentiality is provided by encrypting the body and signature using the Advanced Encryption Standard (AES). This policy set follows the WS-Security and Secure Conversation specifications for issuing and renewing trust operation requests.

To use this policy set, you must also use the Client sample V2 and Provider sample V2 general sample bindings for your applications. For more information, refer to the topic General sample bindings for JAX-WS applications. To use this new policy set, create a new profile after installing the product.

To update existing profiles with this new policy set and the general bindings, Client sample V2 and Provider sample V2 general sample bindings, you must complete some manual steps. You only need to update the deployment manager profile and stand-alone application server profiles. To complete the manual steps for an existing profile, refer to the topic Configuring Kerberos policy sets and V2 general sample bindings.

WSReliableMessaging default

This policy set enables both WS-ReliableMessaging Version 1.1 and WS-Addressing and uses the minimum quality of service, *unmanaged non-persistent*. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages. In-order delivery is set to “false”, so messages are not necessarily delivered in the order in which they were sent.

WSReliableMessaging persistent

This policy set enables both WS-ReliableMessaging and WS-Addressing and uses the maximum quality of service, *managed persistent*. This quality of service supports asynchronous web service invocations and uses a service integration messaging engine and message store to manage the sequence state. Messages are processed within transactions, are persisted at the web service requester server and at the web service provider server, and are recoverable in the event of server failure. In-order delivery is set to “false”, so messages are not necessarily delivered in the order in which they were sent.

Because this policy set specifies managed persistent quality of service, you have to define bindings to the service integration bus and messaging engine that you want to use to manage the WS-ReliableMessaging state. You can attach and bind a WS-ReliableMessaging policy set to a web service application by using the administrative console or the wsadmin tool.

WSReliableMessaging 1_0

This policy set enables both WS-ReliableMessaging Version 1.0 and WS-Addressing and uses the minimum quality of service, *unmanaged non-persistent*. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages. In-order delivery is set to “false”, so messages are not necessarily delivered in the order in which they were sent.

You can use this policy set with .NET-based web services.

WSSecurity default

This policy set provides:

- Message integrity through digital signature (using RSA public-key cryptography) to sign the body, time stamp, and WS-Addressing headers using WS-Security specifications.
- Message confidentiality through encryption (using RSA public-key cryptography) to encrypt the body, signature and signature elements using WS-Security specifications.

LTPA WSSecurity default

This policy set provides:

- Message integrity through digital signature (using RSA public-key cryptography) to sign the body, time stamp, and WS-Addressing headers using WS-Security specifications.
- Message confidentiality through encryption (using RSA public-key cryptography) to encrypt the body, signature and signature elements using WS-Security specifications.
- A Lightweight Third Party Authentication (LTPA) token included in the request message to authenticate the client to the service.

Username WSSecurity default

This policy set provides:

- Message integrity through digital signature (using RSA public-key cryptography) to sign the body, time stamp, and WS-Addressing headers using WS-Security specifications.

- Message confidentiality through encryption (using RSA public-key cryptography) to encrypt the body, signature and signature elements using WS-Security specifications.
- A username token included in the request message to authenticate the client to the service. The username token is encrypted in the request.

WSTransaction

This policy set enables WS-Transaction, which provides:

- The ability to coordinate distributed transactional work atomically and interoperably using the WS-AtomicTransaction specification.
- The ability to coordinate loosely coupled business processes that are distributed across the heterogenous web service environment, with the ability to compensate actions if a failure occurs in the business activity, using the WS-BusinessActivity specification.

SSL WSTransaction

This policy set enables WS-Transaction, which provides:

- The ability to coordinate distributed transactional work atomically, interoperably, and securely, using the WS-AtomicTransaction specification and SSL Transport security.
- The ability to coordinate loosely coupled business processes, with the ability to compensate actions if a failure occurs in the business activity, securely, using the WS-BusinessActivity specification and SSL Transport security.

Policy sets do not include environment or platform-specific information, such as keys for signing, keystore information, or persistent store information. This type of information is defined in the binding. A policy set attachment defines how a policy set is attached to service resources and bindings. The attachment definition is outside the policy set definition and is defined as meta-data associated with application data.

Bindings are made up of environment and platform-specific information. General bindings such as the service client or provider bindings for the global security domain can be shared across applications.

To enable policy sets to work with applications, bindings are needed. Use the administrative console to configure general bindings and application specific bindings. Read about defining binding information for policy sets for more information about working with attachments and bindings.

Overview of migrating policy sets and bindings

Policy sets are migrated during the product migration from Version 6.1 Feature Pack for Web Services or Version 7.0 to Version 8.0. This topic describes the different rules that apply to migrating policy sets and bindings. For information about migrating the version of the product that you are running, see migrating and coexisting.

Migration from Version 6.1 Feature Pack for Web Services

Migration rules for policy sets

For the following policy sets, the migration code first checks for any application attachment for a policy set. If there is an application attachment, the migration code upgrades and renames the policy set to *oldPolicySet_wsfep*; then updates the policy set attachments from *oldPolicySet* to *oldPolicySet_wsfep* in Version 7.0 and later. If there is no attachment for the policy set, then the policy set is not migrated or renamed.

- LTPA RAMP default
- LTPA SecureConversation
- LTPA WSSecurity default
- Username RAMP default
- Username SecureConversation
- Username WSSecurity default

Other policy sets are migrated to Version 7.0 and later only if there is no other policy set in Version 7.0 and later with the same name.

Migration rules for bindings

In Version 7.0 and later, custom bindings are called application specific bindings. Read web services policy set bindings.

The following migration rules apply to application specific bindings, cell level default bindings, server level default bindings, and trust service bindings:

Application specific bindings

During migration, the Version 6.1 application specific bindings in the applications are not upgraded. They are copied with the applications to the Version 7.0 and later system. After migration, you can upgrade the Version 6.1 application specific bindings in an application using the administrative command, `upgradeBindings`.

Cell level default bindings

During migration, the cell level default bindings from the Version 6.1 Feature Pack for Web Services are copied to the Version 7.0 and later system, and they replace any existing Version 6.1 cell default bindings in the Version 7.0 and later system. The Version 6.1 cell default bindings are migrated to general bindings in Version 7.0 and later.

Attention:
There is a security fix for the Version 6.1 cell level default WSSecurity bindings. Run the script, *updateBindings.py* that is located in `WAS_HOME/bin` to fix issues with the Version 6.1 cell level default WSSecurity bindings.

You can install a WebSphere Application Server Version 6.1 Feature Pack for Web Services application within the WebSphere Application Server Version 7.0 and later environment. If your application contains Version 6.1 application specific bindings, the application is defined as a Version 6.1 application to the application server. Additionally, if your application is installed on a WebSphere Application Server Version 6.1 Feature Pack for Web Services server within the WebSphere Application Server Version 7.0 and later environment, the application specific binding is created as a Version 6.1 application specific binding.

When you create an application specific binding, the system checks to see if the application is installed on a Version 6.1 server or if the application contains any Version 6.1 application specific bindings. If either of these conditions is true, the system creates Version 6.1 application specific bindings. In addition, you are not allowed to assign a general binding to a policy set attachment for that application because general bindings are for Version 7.0 and later level of the product.

Server level default bindings

During migration, the server default bindings in the Version 6.1 Feature Pack for Web Services are copied over to the Version 7.0 and later server. The server default bindings are also migrated to general bindings. You can decide whether or not to use these general bindings as the Version 7.0 and later server default bindings. The Version 6.1 applications and other applications that are installed on a Version 6.1 server use the Version 6.1 default bindings.

Trust service bindings

Trust service bindings are migrated during the product migration. The migrated trust service bindings replace the trust service bindings in Version 7.0 and later.

Migrating general bindings from Version 7.0 and later to Version 8.0

A general binding is migrated from Version 7.0 to Version 8.0 only if there is no other general binding in Version 8.0 with the same name. If a general binding is not migrated because it has the same name as a Version 8.0 binding, then you can export the binding from the Version 7.0 system and import it into Version 8.0 with a different name.

The default binding settings which specify which provider and client bindings are used as defaults are not migrated from Version 7.0 to Version 8.0. You must modify your default binding settings in Version 8.0 if you want to use the same defaults that were specified in Version 7.0.

Chapter 30. Administering web services - bus-enabled web services

You can associate your web services with the service integration bus, to achieve the following goals: make internal services available as web services; make external web services available internally at bus destinations. Bus-enabled web services also provide a choice of quality of service and message distribution options for web services, along with intelligence in the form of mediations that allow for the rerouting of messages.

Enabling web services through the service integration bus

Web services can use the service integration bus to provide a single point of control, access, and validation of web service requests and allow control of web services that are available to different groups of web service users.

About this task

With bus-enabled web services you can achieve the following goals:

- Create an *inbound service*: Take an internally-hosted service that is available at a bus destination, and make it available as a web service.
- Create an *outbound service*: Take an externally-hosted web service, and make it available internally at a bus destination.

Bus-enabled web services provide a choice of quality of service and message distribution options, along with intelligence in the form of mediations that allow for the rerouting of messages.

To enable web services through service integration technologies, complete the following steps:

Procedure

1. Optional: Learn about bus-enabled web services. Explore the concepts that underly service integration bus-enabled web services.
2. Plan your bus-enabled web services installation. Determine the bus-enabled web services roles that each server is to perform.
3. Ensure that every server that is to play a bus-enabled web services role is a member of a service integration bus. For more information, see “Configuring the members of a bus” on page 1889.
4. For every server that is to play a bus-enabled web services role, install and configure a Service Data Objects (SDO) repository on the server.

   Note: For WebSphere Application Server Version 6.0, you also had to manually install a selection of the following applications:
   - The service integration technologies resource adapter (used to invoke web services at outbound ports).
   - The bus-enabled web services application.
   - One or more endpoint listener applications.

   For later versions of WebSphere Application Server, these applications are installed automatically as and when needed. For example, the endpoint listener application is installed automatically when you configure an endpoint listener.
5. Create a new endpoint listener configuration for each endpoint listener application that you plan to use to receive inbound service requests.
6. Optional: Create an inbound service. An inbound service is a web interface to a service that is provided internally (that is, a service provided by your own organization and hosted in a location that is

directly available through a service integration bus destination). To configure a locally-hosted service as an inbound service, you associate it with a service destination, and with one or more endpoint listeners through which service requests and responses are passed to the service. You can also choose to have the local service made available through one or more UDDI registries.

7. Optional: Create an outbound service. An outbound service is a web service that is hosted externally, and is made available through a service integration bus. To make an externally-hosted service available through a bus, you first associate it with a service destination, then you configure one or more port destinations (one for each type of binding, for example SOAP over HTTP or SOAP over JMS) through which service requests and responses are passed to the external service. You get the port definitions from the WSDL, but you can choose which ones you want to create.
8. Optional: Apply additional security to your bus-enabled web services. By default, the bus-enabled web services configuration works when WebSphere Application Server security is enabled and your service integration buses are secured. However this level of security does not impose any security restrictions on the users of your bus-enabled web services configuration. To control how your bus-enabled web services configuration is used by each group of your colleagues or customers, use the bus-enabled web services additional security features to enable working with password-protected components and servers, with WS-Security and with HTTPS.

What to do next

For more information about specific aspects of bus-enabled web services, see the following topics:

- Bus-enabled web services
- "Installing and configuring the SDO repository"
- "Configuring web services for a service integration bus" on page 2777
- "Administering the bus-enabled web services resources" on page 2789
- Securing bus-enabled web services
- "Passing SOAP messages with attachments through the service integration bus" on page 2824
- "SIBWebServices command group for the AdminTask object" on page 2933
- Bus-enabled web services troubleshooting tips
- Tuning bus-enabled web services

Installing and configuring the SDO repository

Service Data Objects (SDO) is an open standard for enabling applications to handle data from different data sources in a uniform way, as data graphs. Service integration bus-enabled web services use an SDO repository for storing and serving WSDL definitions. Use this task to create and configure your preferred database to store SDO data, and to install and configure an SDO repository on each server that you plan to use for bus-enabled web services.

Before you begin

Determine the servers on which to install and configure an SDO repository as described in Planning your bus-enabled web services installation, then add each server as a member of a bus as described in "Configuring the members of a bus" on page 1889.

An SDO repository can work with most database products. For specific information about choosing and configuring your preferred database, consult your database administrator or database product documentation, and read the notes in this topic on database usage.

About this task

To install and configure an SDO repository, complete the following steps:

- Install your preferred database product.

- Create a JDBC provider and a data source for your database.
- Run the `installSdoRepository.jacl` script one or more times, to install the SDO application on each server and to set the database type that the SDO repository is to use.

For more information about how to do this, first read the following notes on database usage and on the `installSdoRepository.jacl` script, and then complete the steps for one of these configurations:

- "Configure the SDO repository for a single server, and to use the embedded Derby database" on page 2774.
- "Configure the SDO repository for a single server, and to use a database other than embedded Derby" on page 2775.

Notes on database usage:

- For a single server configuration, you can use either your preferred database or the embedded Apache Derby database that is supplied with WebSphere Application Server.
- The SDO repository dictates the schema and table names that it uses, so different repositories must use different databases to ensure that they do not access the same data.
- Create the database for your preferred database supplier by using the `Table.ddl` file from the relevant *app_server_root*`/util/SdoRepository/`*database_type* directory. The `Table.ddl` file describes the database table that is needed by the SDO repository.
- The **-editBackendId** flag on the `installSdoRepository.jacl` script determines the database type that the repository is to use. The back end ID determines what database-specific rules the application follows when talking to the database. See the associated note on the `installSdoRepository.jacl` script.
- Some databases require a user ID that has been granted permissions to access the SDO repository database. Create a user ID for user name `SDOREP` before you create the tables for Oracle, Sybase, and SQL Server databases. Because of the way these databases handle user names and table names, the user name must be `SDOREP` to enable the SDO repository to access its table with the fully qualified name `SDOREP.BYTESTORE`. Make sure that you grant permission for the `SDOREP` user to read from, and write to, the database.
- If you use an Informix database, do not disable logging.
- The SDO repository does not require *XA* support. In most cases you can use either an XA or a non-XA data source. However, if your database is Oracle 8 or 9, you must use the Oracle JDBC driver (non-XA) for the SDO repository data source.
- You might also choose to complete other steps such as creating an index of the primary key to improve database performance. Do not change the schema, table and column names.

Notes on the installSdoRepository.jacl script:

- Use the wsadmin scripting client to run the script.
- Run the script from within QShell.
- The script is provided in the *app_server_root*`/bin` directory, where *app_server_root* is the root directory for the installation of WebSphere Application Server. If you choose to run the wsadmin scripting client from another directory, specify the full path to the script on

the command option. For example to work with a profile other than the default profile, change to the *app_server_root*/profiles/*profile_name*/bin directory then specify the following path to the script:

```
wsadmin -f app_server_root/bin/installSdoRepository.jacl
```

- The **-editBackendId** flag on the installSdoRepository.jacl script determines the database type that the repository is to use. The back end ID determines what database-specific rules the application follows when talking to the database. To see the full list of available back end ID values, use the -listBackendIds flag:

  ```
  wsadmin -f installSdoRepository.jacl -listBackendIds
  ```

  All the back end ID values in the list can be used when the SDO repository is installed on one or more WebSphere Application Server Version 7.0 or later application servers. Values marked with (*) cannot be used when the SDO repository is installed on Version 6.0 servers. Values marked with (**) cannot be used when the SDO repository is installed on Version 6.0 or Version 6.1 servers.
- If the data source already exists, or there has been a previous broken or partial installation of the SDO repository application, the installSdoRepository.jacl script fails to complete and configuration changes are not saved. In these cases, run the SDO repository uninstall script, fix the problem, then rerun the installSdoRepository.jacl script.

Configure the SDO repository for a single server, and to use the embedded Derby database

About this task

If you are creating a single server configuration and you want to use embedded Derby, you run the installSdoRepository.jacl script with the -createDb switch. This action creates the Derby database and installs the SDO repository.

To configure the SDO repository for a single server and to use the embedded Derby database, complete the following steps:

Procedure

1. Open a command prompt, then change to the *app_server_root*/bin directory.
2. Enter the following command:

   ```
   wsadmin -f installSdoRepository.jacl -createDb
   ```

   Note: The -createDb flag tells the command to create a default Derby database. If you omit this flag, the command still installs an SDO repository that is configured to use Derby, but the command does not also create the database.

Configure the SDO repository for a single server, and to use a database other than embedded Derby

About this task

If you are creating a single server configuration that uses a database other than embedded Derby, you install your preferred database product, then create a JDBC provider and a data source, then run the `installSdoRepository.jacl` script twice:

1. One time to install the SDO application on the application server.
2. One time to set the database type that the SDO repository is to use.

To configure the SDO repository for a single server and to use a database other than embedded Derby, complete the following steps:

Procedure

1. Create the database for your preferred database supplier by using the `Table.ddl` file from the relevant *app_server_root*`/util/SdoRepository/`*database_type* directory.

   For an illustration of the process for creating tables in DB2, see Recreating database tables from the exported table data definition language . For more information, see “Deploying data access applications” on page 171.
2. Create a J2C authentication alias.

   This is for use with the data source that you create in the next step. Check that the authentication alias matches the login details for your database instance, otherwise a connection will not be made.
3. Create and configure a JDBC provider and data source.

   Set the following data source properties:
   - Set the **authentication** property to use the authentication alias you created in the previous step.
   - Select the **Use this Data Source in container managed persistence (CMP)** check box.
   - Set the **Name** property to a name of your own choosing. For example, SDO Repository DataSource.
   - Set the **JNDI name** property to the following exact value: jdbc/com.ibm.ws.sdo.config/SdoRepository.
   - Set any other properties that are required settings for your chosen database.
4. Optional: Test the data source connection:

   Note: This option does not work in all configurations. The availability of this option depends on the scope at which the data source is defined, and the scope of any WebSphere Application Server variables that are used in the JDBC provider and data source configurations. For more information about testing connections to data sources, see Test connection service.

   a. In the administrative console, navigate to **Resources -> JDBC -> Data sources**.
   b. Select the SDO repository data source.
   c. Click **Test connection**.
5. Configure the SDO repository:
   a. Open a command prompt, then change to the *app_server_root*`/bin` directory.
   b. Install the SDO repository application on the server:

      ```
      wsadmin -f installSdoRepository.jacl
      ```
   c. Set the database type that the SDO repository is to use:

```
wsadmin -f installSdoRepository.jacl -editBackendId database_type
```

      for example:

```
wsadmin -f installSdoRepository.jacl -editBackendId DB2UDB_V82
```

The SDO repository uninstall script

Use this script to uninstall a Service Data Objects (SDO) repository that was previously installed, or failed to install correctly.

You install the SDO repository application on every server that you plan to use for one or more of the service integration bus-enabled web services roles as described in “Installing and configuring the SDO repository” on page 2772.

If the data source already exists, or there has been a previous broken or partial installation of the SDO repository application, the `installSdoRepository.jacl` script fails to complete and configuration changes are not saved. In these cases, you have to run the `uninstallSdoRepository.jacl` script. This script continues when it finds unexpected results, so it can clean up a broken or partial installation.

Note: Run the script from within QShell.

The script is provided in the *app_server_root*`/bin` directory, where *app_server_root* is the root directory for the installation of WebSphere Application Server. If you choose to run the wsadmin scripting client from another directory, specify the full path to the script on the command option. For example to work with a profile other than the default profile, change to the *app_server_root*`/profiles/`*profile_name*`/bin` directory then specify the following path to the script:

```
wsadmin -f app_server_root/bin/uninstallSdoRepository.jacl
```

The SDO repository script install and uninstall pairs

The following are the install and uninstall command pairs, where each uninstall command undoes the action of the related install command. If you attempt to uninstall with a different set of arguments to those previously used with the `installSdoRepository.jacl` script, you might find that the uninstall does not remove everything or that it displays warnings when it tries to remove non-existent settings.

For configuration of the SDO repository on a server, the `-createDb` flag tells the install command to create a default (Apache Derby) database and configure it for use with this application server. The `-removeDb` flag tells the uninstall command to remove the database configuration from the application server, but not to delete the Apache Derby database:

```
wsadmin -f installSdoRepository.jacl -createDb
wsadmin -f uninstallSdoRepository.jacl -removeDb
```

Note:

- If you did not use `-createDb` on the installer, because you had already configured an Apache Derby database for some other purpose, then you should not use the `-removeDb` flag on the uninstaller.
- To avoid deleting data that you might want to keep, the `-removeDb` flag does not delete the Apache Derby database. If you are certain that you want to delete the database, you can do so manually. An Apache Derby database is a directory on the file system. The one created by the installer with the `-createDb` flag is in the *profile_root*`/databases/SdoRepDb` directory, where *profile_root* is the directory in which profile-specific information is stored. If you do not delete the database, and you try to install again with the `-createDb` flag, the installation process fails stating that the `SdoRepDb` directory already exists.

For installation or removal of the SDO repository application from a server:

```
wsadmin -f installSdoRepository.jacl
wsadmin -f uninstallSdoRepository.jacl
```

Bus-enabled web services installation files and locations

When you install WebSphere Application Server, the files that are required for running service integration bus-enabled web services are copied into your file system under *app_server_root*, where *app_server_root* is the root directory for the installation of WebSphere Application Server.

The following table lists the main bus-enabled web services files that you might have to access, and the locations into which they are placed.

Table 259. Bus-enabled web services files and their locations. The first column of this table lists the main bus-enabled web services files, the second column describes the purpose of the files, the third column lists the locations of the files.

File name	Purpose	Location
`installSdoRepository.jacl`	The script used to configure the SDO repository. Bus-enabled web services use an SDO repository for storing and serving their WSDL definitions.	*app_server_root*`/bin`
`uninstallSdoRepository.jacl`	The script used to uninstall the SDO repository.	*app_server_root*`/bin`
SDO repository resources	The resource files used to configure the SDO repository to work with a range of different databases.	*app_server_root*`/util/SdoRepository`
`sibwsauthbean.ear`	The application for Password-protecting a web service operation.	*app_server_root*`/installableApps`
`sibwsAuthGen`	The command file used to generate authorization beans for password-protecting a web service operation.	*app_server_root*`/util`
`soaphttpchannel1.ear`	The SOAP over HTTP endpoint listener 1 application. This application is installed automatically when you create an associated endpoint listener configuration.	*app_server_root*`/installableApps`
`soaphttpchannel2.ear`	The SOAP over HTTP endpoint listener 2 application. This application is installed automatically when you create an associated endpoint listener configuration.	*app_server_root*`/installableApps`
`soapjmschannel1.ear`	The synchronous SOAP over Java Messaging Service (JMS) endpoint listener 1 application. This application is installed automatically when you create an associated endpoint listener configuration.	*app_server_root*`/installableApps`
`soapjmschannel2.ear`	The synchronous SOAP over JMS endpoint listener 2 application. This application is installed automatically when you create an associated endpoint listener configuration.	*app_server_root*`/installableApps`

Configuring web services for a service integration bus

Take an internally-hosted service that is available at a bus destination, and make it available as a web service; Take an externally-hosted web service, and make it available internally at a bus destination; Use the web services gateway to map an existing service - either an inbound or an outbound service - to a new Web service that appears to be provided by the gateway.

About this task

In figure 1, a client request is received by an endpoint listener then passed through an inbound port to an inbound service destination. An outbound service destination passes a request through an outbound port to an external service.

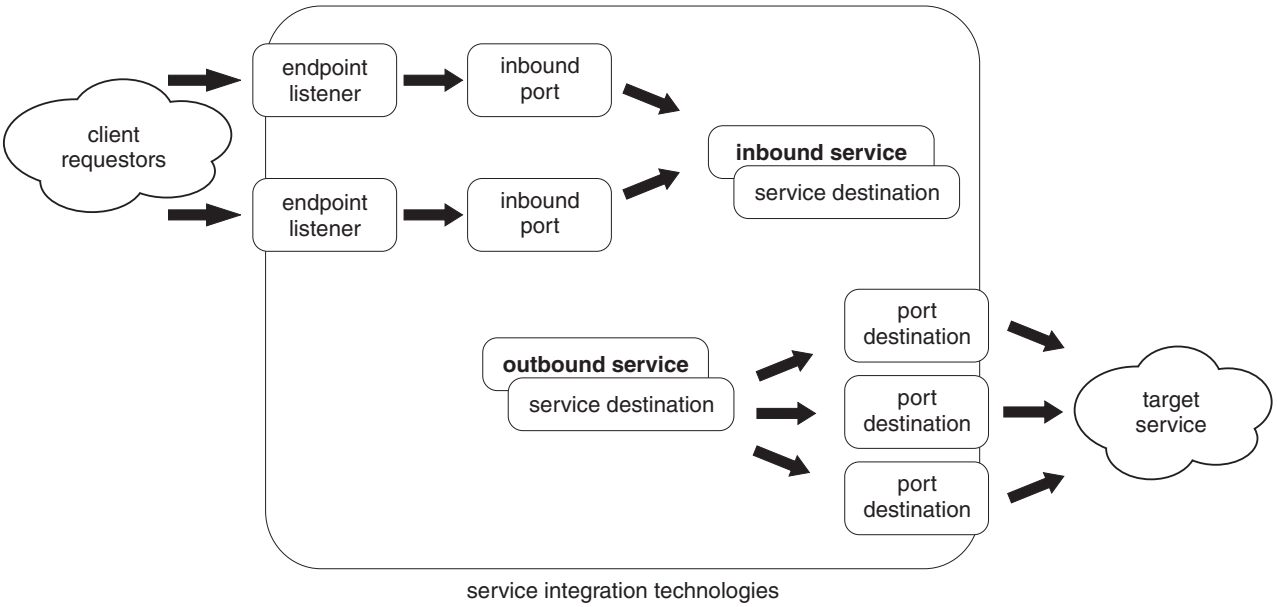


Figure 42. Inbound and outbound services

In figure 2, a client request is received by an endpoint listener, then passed through an inbound port to an inbound service destination. An outbound service destination passes a request through an outbound port to an external service, and a gateway service resembles an inbound service and maps to an outbound service.

Through service integration bus-enabled web services you can achieve the following goals:

Procedure

- Create an *inbound service*: Take an internally-hosted service that is available at a bus destination, and make it available as a web service.
- Create an *outbound service*: Take an externally-hosted web service, and make it available internally at a bus destination.

Making an internally-hosted service available as a web service

Create an inbound service. An inbound service is a Web interface to a service that is provided internally (that is, a service provided by your own organization and hosted in a location that is directly available through a service integration bus destination). To configure a locally-hosted service as an inbound service, you associate it with a service destination, and with one or more endpoint listeners through which service requests and responses are passed to the service. You can also choose to have the local service made available through one or more UDDI registries.

Before you begin

This topic assumes that:

- You have created and installed a Service Data Objects (SDO) repository (used for storing and serving WSDL definitions) on every server that is to play a service integration bus-enabled web services role.
- You have created a new endpoint listener configuration for each endpoint listener that you plan to use to receive inbound service requests.
- You already have an internally-hosted service that you want to configure as an inbound service, and you have made the service available at a service integration bus destination.
- You have created references to any UDDI registries in which you want to register this service.

You must also create a template WSDL file that describes the service, and make the WSDL available at a URL or through a UDDI registry. For information on how to create a WSDL file, see Developing a WSDL file for JAX-RPC applications.

You can create an inbound service by using the administrative console as described in this task, or by using the “createSIBWSInboundService command” on page 2940.

Note: If the bus needs to pass messages through an authenticating proxy server to retrieve WSDL documents, then you cannot use the administrative console for this task and you must create your new inbound service by using the wsadmin tool. For more information see the corresponding troubleshooting tip.

About this task

In the following figure, a client request is received by an endpoint listener, then passed through an inbound port to an inbound service destination. JAX-RPC handlers and WS-Security bindings can be applied at the ports.

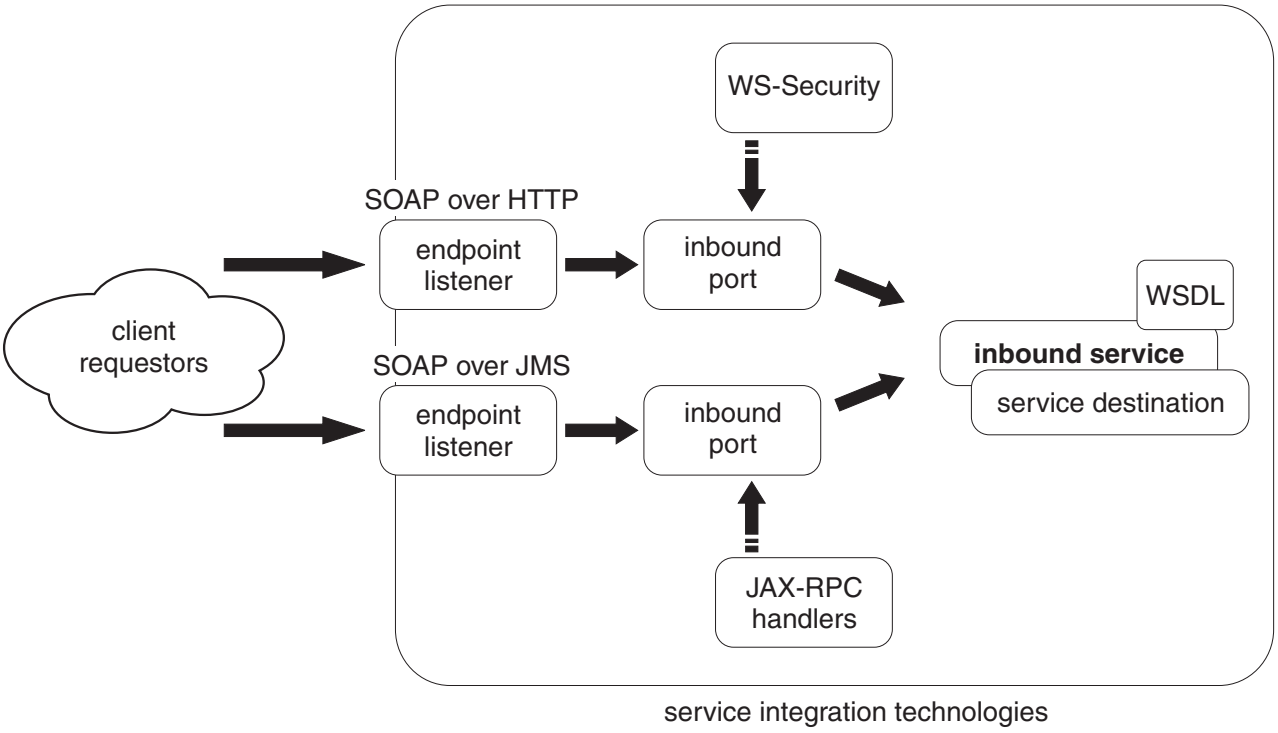


Figure 43. Inbound service

Web service requests and responses to an inbound service can be sent across any binding (for example SOAP over HTTP or SOAP over JMS) that is available to the bus. Each available binding type is represented by an inbound port, and each inbound port is associated with a binding-specific endpoint listener.

You can control and monitor access to your inbound services in the following ways:

- You can control which groups of users can access a particular inbound web service by making the service available only through specific endpoint listeners.
- You can associate JAX-RPC handler lists with ports, so that the handlers can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.

- You can set the level of security to be applied to messages (the WS-Security configuration and bindings). The security level can be set independently for request and response messages.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Services] Inbound Services**. The inbound services collection form is displayed.
3. Click **New**. The New inbound service wizard is displayed.
4. Use the wizard to create the new inbound service configuration by completing the following steps. For more information about the properties that you set with the wizard, see Inbound services settings.
   a. Select the service destination and template WSDL location.

      Note: The template WSDL is the service-specific WSDL file that you have created to describe this inbound service.

   b. Select the service from the template WSDL.

      Note:

      - This option is needed in case there is more than one service in the template WSDL. The field is filled in for you by default. If there is only one service in the WSDL, accept the default.

   c. Specify the name of the inbound service and select the endpoint listeners.

      Note:

      - You need not supply a name for the inbound service. If you choose not to supply a name, a default name is created. The default name is derived from the service destination name, with characters that are not valid for names filtered out.
      - An inbound port is automatically created for each endpoint listener that you select. Each inbound port is created without a template port, JAX-RPC handler list or security settings and is given a default name that relates to the endpoint listener selected. For an overview of the relationship between endpoint listeners and inbound ports, see Endpoint listeners and inbound ports: Entry points to the service integration bus.

   d. Define any UDDI publication properties.

      Note: You can use the wizard to specify the UDDI publication properties that are used to publish this inbound service to an initial UDDI registry. After you create an inbound service through the wizard, you can use the modify an existing inbound service configuration option to publish the service to more UDDI registries. For information about the UDDI publication properties, see UDDI Publication settings and UDDI registries: Web service directories that can be referenced by bus-enabled web services.
5. Click **Finish**.

Results

If the processing completes successfully, the list of inbound services for this service integration bus is updated to include the new inbound service. Otherwise, an error message is displayed.

What to do next

If you want to secure your new inbound service, or apply any JAX-RPC handler lists to the ports for the service, or publish the service to more UDDI registries, use the administrative console to modify your inbound service configuration.

Modifying an existing inbound service configuration:

Modify the configuration details for an existing inbound service. For example: secure the service; apply JAX-RPC handler lists to the ports for the service; publish the service to more than one UDDI registry.

About this task

An inbound service is a web interface to a service that is provided internally (that is, a service provided by your own organization and hosted in a location that is directly available through a service integration bus destination).

When you first create an inbound service, you connect the service to one or more endpoint listeners and (optionally) specify the UDDI publication properties that are used to publish the inbound service to an initial UDDI registry. An inbound port is automatically created for each endpoint listener that you select, but each inbound port is created without a template port, JAX-RPC handler list or security settings. Modify your inbound service configuration if you want to control and monitor access to your inbound services in any of the following ways:

- Associate JAX-RPC handler lists with ports, so that the handlers can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.
- Password-protect a web service operation.
- Set the level of security to be applied to messages (the WS-Security configuration and bindings). The security level can be set independently for request and response messages.
- Publish the service to more than one UDDI registry.

To list the existing inbound services, and to view and modify their configuration details, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Services] Inbound Services**. A list of all the inbound services is displayed in an inbound services collection form.
3. Click the name of an inbound service in the list. The current settings for this inbound service are displayed.
4. Optional: Click **Reload template WSDL** to reload the template WSDL file for this inbound service.

   Note:

   - When you create a new inbound service, a copy of the template WSDL file for the service is loaded into a locally-maintained repository. If you change the template WSDL file, you must update the local copy.
   - When you click **Reload template WSDL**, you run the command that is described in Refreshing the inbound service WSDL file by using the wsadmin tool. For the command to complete successfully, the conditions must be met that are described in that topic.
   - If the bus needs to pass messages through an authenticating proxy server to retrieve WSDL documents, then you cannot use the **Reload template WSDL** option and you must run the refresh WSDL command by using the wsadmin tool. For more information, see the corresponding troubleshooting tip.
5. Modify the general properties. For information about each of these properties, see Inbound services settings.

   Note:

   - When you change an inbound service name, the system looks up all objects that refer to it and updates the name.

- The template WSDL is the service-specific WSDL file that you create to describe this inbound service. For information about how to create a WSDL file, see Developing a WSDL file for JAX-RPC applications.
- Although logically the template WSDL name and namespace are only required if there is more than one service in the WSDL, the fields that you use to set them are coded within the administrative console as compulsory fields. They are filled in for you by default, so if they are not logically required for your service you should leave the default values. If you remove the value from either field, the administrative console treats the empty field as an error.
- If you select the option to **Enable operation-level security** then you must also complete, for this inbound service, the steps described in Password-protecting a web service operation.

6. Modify the additional properties.
   a. Modify the inbound ports that are associated with this inbound service.

      An inbound port describes the web service enablement of a service destination on a specific endpoint listener, with associated configuration. Each inbound port is associated with an endpoint listener, and you can control which groups of users can access a particular inbound service by making the service available only through specific endpoint listeners. For more information, see Endpoint listeners and inbound ports: Entry points to the service integration bus.

      You can use a JAX-RPC handler list to monitor activity at the port, and take appropriate action (for example logging, or rerouting) depending upon the sender and content of each message that passes through the port. For more information, see Bus-enabled web services and JAX-RPC handlers.

      You can use WS-Security to set the levels of security to be applied to messages. The security level can be set independently for request and response messages. For more information, see Service integration technologies and WS-Security.

      See also Inbound ports settings.

   b. Modify the UDDI publication properties that are used to publish this inbound service to one or more UDDI registries. For information about the UDDI publication properties, see UDDI Publication settings and UDDI registries: Web service directories that can be referenced by bus-enabled web services.

   c. Modify the custom properties, if any, that you have set for this inbound service. These custom properties are name and value pairs that you can use to set internal system configuration properties. In each pair, the name is a property key and the value is a string value.

   d. Use the publish WSDL files property to export the template WSDL for this inbound service to a compressed file.

      As a technology preview, the exported compressed file includes a version of the WSDL file that has no ports (bindings) defined. This non-bound WSDL is intended for use by your colleagues preparing to deploy an inbound service. It gives you a convenient way of sharing information about the planned deployment details for the service among your team. When you finally deploy the inbound service, the associated WSDL must be complete (that is, it must include the binding information).

      The non-bound WSDL file is always published in the exported compressed file for the inbound service, along with the bound WSDL file if the inbound service has any ports defined. The compressed file, named *inbound_service_name*.zip, therefore always contains the following files:

      - *bus_name*.*inbound_service_name*NonBound.wsdl (this file contains the non-bound service, port and binding for the inbound service).
      - *bus_name*.*inbound_service_name*PortTypes.wsdl (this file contains the port type definition for the inbound service).

      If the inbound service has one or more ports, then the compressed file additionally contains the following files:

      - *bus_name*.*inbound_service_name*Service.wsdl (this file contains the service and port elements for the inbound service).

- *bus_name*.*inbound_service_name*Bindings.wsdl (this file contains the binding elements that correspond to the ports for the inbound service).

If there is an error generating the WSDL then an error page is returned.

7. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of inbound services for this service integration bus is redisplayed. Otherwise, an error message is displayed.

Deleting inbound services configurations:

Use this task to delete inbound services configurations using the administrative console.

Before you begin

Decide which method to use to configure these resources. You can delete an inbound service by using the administrative console as described in this task, or by using the “deleteSIBWSInboundService command” on page 2941.

About this task

To delete one or more inbound services by using the administrative console, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Services] Inbound Services**. A list of all the inbound services is displayed in an inbound services collection form.
3. Select the check box for every inbound service that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of inbound services for this service integration bus is updated. Otherwise, an error message is displayed.

Making an externally-hosted web service available internally

Create an outbound service. An outbound service provides access, through one or more outbound ports, to a web service that is hosted externally. An outbound service can be used by any of your internal systems that can access the service integration bus on which it is hosted. To make an externally-hosted service available through a bus, you first associate it with a service destination, then you configure one or more port destinations (one for each type of binding, for example SOAP over HTTP or SOAP over JMS) through which service requests and responses are passed to the external service. You get the port definitions from the WSDL, but you can choose which ones you want to create.

Before you begin

This topic assumes that you have created and installed a Service Data Objects (SDO) repository (used for storing and serving WSDL definitions) on every server that is to play a service integration bus web services role.

To create an outbound service, you must know the location of the externally-published WSDL file that describes the service. This WSDL file is either available at a web address or through a UDDI registry.

If the WSDL file for your outbound service is stored in a UDDI registry, you associate the outbound service with a UDDI reference to the registry. You select the UDDI reference from a drop-down list, so you must configure the UDDI reference before you configure a new outbound service that uses it.

Decide which method to use to configure these resources. You can create an outbound service by using the administrative console as described in this task, or by using the “createSIBWSOutboundService command” on page 2934.

Note: If the bus needs to pass messages through an authenticating proxy server to retrieve WSDL documents, then you cannot use the administrative console for this task and you must create your new outbound service by using the wsadmin tool. For more information see the corresponding troubleshooting tip.

About this task

In the following figure, each message is passed from the outbound service to the target service through an outbound port. A separate outbound port is created for each available binding. JAX-RPC handlers and WS-Security settings can be applied at the ports.

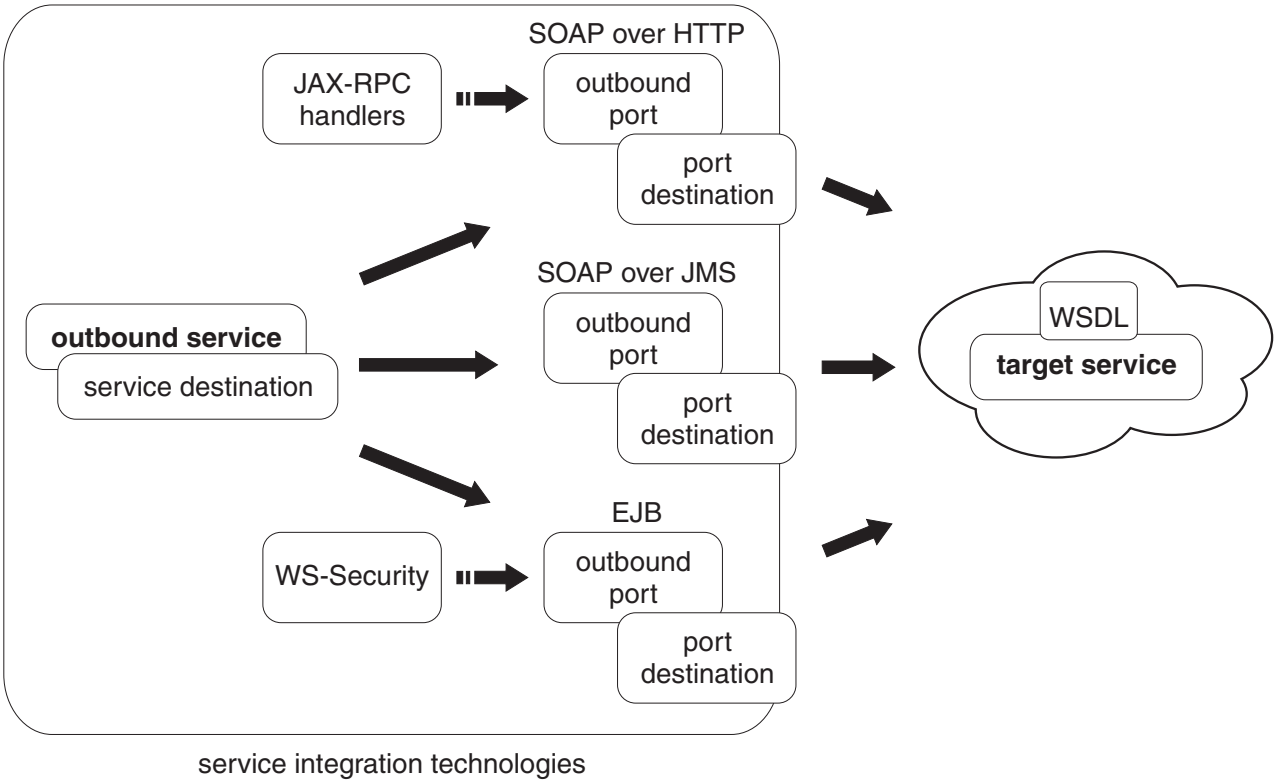


Figure 44. Outbound service

Requests and responses to an outbound service are sent across any transport binding (for example SOAP over HTTP, SOAP over JMS, EJB binding) that is available to both the target service and the service integration bus. Each available binding type is represented by an outbound port configured at a port destination. For more information, see Outbound ports and port destinations.

You can control and monitor access to the target service in the following ways:

- You can associate JAX-RPC handler lists with ports, so that the handlers can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.

- You can set the level of security to be applied to messages (the WS-Security binding). The security level can be set independently for request and response messages.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Services] Outbound Services**. The outbound services collection form is displayed.
3. Click **New**. The New outbound service wizard is displayed.
4. Use the wizard to create the new outbound service configuration by completing the following steps. For more information about the properties that you set with the wizard, see Outbound services settings and Outbound ports settings.
   a. Locate the target service WSDL.
   b. Select the service from the WSDL.

      Note:
      - This option is needed in case there is more than one service in the WSDL. The field is filled in for you by default. If there is only one service in the WSDL, accept the default.
      - There needs to be at least one port defined in the service you select.

   c. Select the ports that are to be enabled for this service.

      Note: Select at least one port.

   d. Name the outbound service, the service destination and all of the port destinations.

      Note:
      - Default names are generated, but you can rename them. The default names are unique within the current service integration bus. Any replacement names that you choose must be similarly unique. If you enter a name that is not unique, an error message is displayed.
      - If you have created a port selection mediation and deployed it to the service integration bus, then it is available for selection in the list of mediations. If you do not want to use a port selection mediation with this outbound service, select `none` from the drop-down list. This list contains all mediations, including port selection mediations, that are currently deployed to this service integration bus.
      - The list of available ports is a subset of the ports that are described in the WSDL file. You chose this subset in the previous step. If you selected more than one port in the previous step, you should also set the default port to be used unless otherwise specified by a port selection mediation.

   e. Assign each port destination and (optionally) the port selection mediation to a bus member.

      Note:
      - Bus members are application servers or clusters that are added to this bus.
      - The option to assign a port selection mediation to a bus member is only displayed if you selected a mediation in the previous step.
5. Click **Finish**.

Results

If the processing completes successfully, the list of outbound services for this service integration bus is updated to include the new outbound service. Otherwise, an error message is displayed.

What to do next

Because the service is hosted externally, you might also need to enable proxy server authentication for each port to get permission to access the Internet.

If you want to secure your new outbound service, or apply any JAX-RPC handler lists to the ports, or enable proxy server authentication for any of the ports, use the administrative console to modify your outbound service configuration.

Modifying an existing outbound service configuration:

Modify the configuration details for an outbound service. For example: secure the service; apply JAX-RPC handler lists to the ports for the service; publish the service to more than one UDDI registry.

About this task

An outbound service provides access, through one or more outbound ports, to a web service that is hosted externally. An outbound service can be used by any of your internal systems that can access the service integration bus on which it is hosted.

When you first create an outbound service you select the ports that are to be enabled for the service, but you do not associate the ports with JAX-RPC handler lists or security settings. You need to modify your outbound service configuration if you want to control and monitor access to the target service in any of the following ways:

- Associate JAX-RPC handler lists with ports, so that the handlers can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.
- Password-protect a web service operation.
- Set the level of security to be applied to messages (the WS-Security binding). The security level can be set independently for request and response messages.
- Enable proxy server authentication for any of the ports.

To list the existing outbound services, and to view and modify their configuration details, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Services] Outbound Services**. A list of outbound services is displayed in an outbound services collection form.
3. Click the name of an outbound service in the list. The current settings for this outbound service are displayed.
4. Optional: Click **Reload WSDL** to reload the external WSDL file for this outbound service.

   Note:

   - When you create a new outbound service, a copy of the external WSDL file for the service is loaded into a locally-maintained repository. If the external service provider changes the WSDL file, you must update the local copy.
   - When you click **Reload WSDL**, you run the command that is described in Refreshing the outbound service WSDL file by using the wsadmin tool. For the command to complete successfully, the conditions must be met that are described in that topic.

- If the bus needs to pass messages through an authenticating proxy server to retrieve WSDL documents, then you cannot use the **Reload WSDL** option and you must run the refresh WSDL command by using the wsadmin tool. For more information see the corresponding troubleshooting tip.

5. Modify the general properties. For information about each of these properties, see Outbound services settings.

   Note:

   - When you change an outbound service name, the system looks up all objects that refer to it and updates the name. Any replacement name that you choose must be unique within the current service integration bus. If you enter a name that is not unique, an error message is displayed.
   - You cannot change the **Service destination name**. However, if you click **View** alongside the name, you can view and modify the configuration information for the service destination.
   - If you change the WSDL location information (that is the fields **WSDL location type**, **WSDL location** and **WSDL UDDI Registry**), then click **Apply**, the outbound service WSDL file is reloaded. Therefore you should click **Apply** after you make any changes to the WSDL location information and before you change any of the WSDL-derived fields (for example WSDL service name, and list of available ports).
   - Although logically the WSDL service name and namespace are only required if there is more than one service in the WSDL, the fields that you use to set them are coded within the administrative console as compulsory fields. They are filled in for you by default, so if they are not logically required for your service you should leave the default values. If you remove the value from either field, the administrative console treats the empty field as an error.
   - The list of available ports from which you choose the **Default port name** is a subset of the ports that are described in the WSDL file. You chose this subset when you created or last modified this outbound service. To add or remove available ports, use the additional properties option **Outbound Ports**.
   - If you have created a port selection mediation and deployed it to the service integration bus, then it is available for selection in the list of mediations. If you do not want to use a port selection mediation with this outbound service, select `none` from the drop-down list. This list contains all mediations, including port selection mediations, that are currently deployed to this service integration bus.
   - Bus members are application servers or clusters that are added to this bus. The **Bus member** property defines the bus member to which the port selection mediation is assigned. If you change the **Port selection mediation** property value to (`none`), you should also change the **Bus member** property value to (`none`). If you want to use a port selection mediation, assign it to a bus member. If you do not do this, the administrative console displays an error message.
   - If you select the option to **Enable operation-level security** then you must also complete, for this outbound service, the steps described in Password-protecting a web service operation.
   -

6. Modify the additional properties.
   a. Modify the ports that are associated with this outbound service. For information about the properties of outbound service ports, see Outbound ports settings.

      Note:

      - Requests and responses to an outbound service can be sent across any binding (for example SOAP over HTTP or SOAP over JMS) that is available to both the service integration bus and the external web service. Each available binding is represented by a port.

- You can use a JAX-RPC handler list to monitor activity at the port, and take appropriate action (for example logging or re-routing) depending upon the sender and content of each message that passes through the port. If the external web service requires HTTP basic authentication, you can use a JAX-RPC handler list to provide an HTTP basic authentication header as described in Invoking a password-protected outbound service.
- You can use WS-Security to set the levels of security to be applied to messages. The security level can be set independently for request and response messages. For more information, see Service integration technologies and WS-Security.
- You can set the levels of security to be applied to messages. The security level can be set independently for request and response messages.
- The service integration technologies require access to the Internet to invoke an outbound service or to retrieve a target service WSDL file. If you use a proxy server in support of Internet routing, and if your proxy server requires authentication before it grants access to the Internet, then you must enable proxy server authentication.

b. Modify the custom properties, if any, that you have set for this outbound service. These custom properties are name and value pairs that you can use to set internal system configuration properties. In each pair, the name is a property key and the value is a string value.

7. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of outbound services for this service integration bus is redisplayed. Otherwise, an error message is displayed.

Deleting outbound service configurations:

Use this task to delete outbound services configurations using the administrative console.

Before you begin

Decide which method to use to configure these resources. You can delete an outbound service by using the administrative console as described in this task, or by using the “deleteSIBWSOutboundService command” on page 2935.

About this task

To delete one or more outbound services by using the administrative console, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Buses -> *bus_name* -> [Services] Outbound Services**. A list of outbound services is displayed in an outbound services collection form.
3. Select the check box for every outbound service that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of outbound services for this service integration bus is updated. Otherwise, an error message is displayed.

Administering the bus-enabled web services resources

Use the administrative console to configure the main service integration bus-enabled web services resources: endpoint listeners; JAX-RPC handler lists; WS-Security bindings and configurations; references to UDDI registries.

About this task

When you configure a bus-enabled web service you associate it with one or more of the following resources:

- The endpoint listeners on which you want the service to be available.
- Any JAX-RPC handler lists that apply to the service.
- Any WS-Security bindings and configurations that apply to the service.
- Any references to UDDI registries in which entries for the service are created.

You choose each of these resources from a list of resources that you have previously configured.

Use the administrative console to administer the bus-enabled web services resources as described in the following topics:

Procedure

1. “Creating a new endpoint listener configuration.”
2. “Working with JAX-RPC handlers and clients” on page 2799.
3. “Working with mediations” on page 2810.
4. “Creating a new UDDI reference” on page 2811.

Creating a new endpoint listener configuration

An endpoint listener is the point (address) at which messages for an inbound service are received. The endpoint listeners that are supplied with WebSphere Application Server support SOAP over HTTP and SOAP over JMS bindings.

Before you begin

For every server that is to host an endpoint listener, you must install and configure a Service Data Objects (SDO) repository on the server.

If you want to change the default HTTP endpoint listener security role, do so before you configure the SOAP over HTTP endpoint listener.

Before you configure a SOAP over JMS endpoint listener, configure the associated JMS resources.

You can set up separate endpoint listeners for inbound and outbound requests. For more information, see Endpoint listeners and inbound ports: Entry points to the service integration bus.

Decide which method to use to configure these resources. You can create a new endpoint listener configuration by using the administrative console as described in this task, or by using the “createSIBWSEndpointListener command” on page 2949.

Note: If you want to create an endpoint listener configuration for your own endpoint listener application, rather than for one of the listeners that is supplied with WebSphere Application Server, you must use the wsadmin tool.

About this task

Endpoint listeners are a physical endpoint for receiving inbound service requests. An inbound service describes a bus destination as a logical web service. An inbound port associates a (logical) inbound

service with an endpoint listener to provide a (physical) endpoint where the service can be invoked. Therefore you must have defined an endpoint listener before you can create an inbound port.

A request arrives at an endpoint listener. It is passed to an inbound port, at which point security and JAX-RPC handler lists can be applied, then sent on to the service destination. Responses follow the same path in reverse.

To configure a new endpoint listener for use with an inbound service is a two-stage process:

1. Configure the listener for a specific application server or cluster (as described in this task).
2. Configure an inbound service on the same bus to use the listener (as described in the task “Making an internally-hosted service available as a web service” on page 2778).

To use the administrative console to configure an endpoint listener, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click one of the following paths:
   - **Servers -> Server Types -> WebSphere application servers -> *server_name* -> Endpoint listeners**
   - **Servers -> Clusters -> WebSphere application server clusters -> *cluster_name* -> Endpoint listeners**

   The endpoint listeners collection form is displayed.
3. Click **New**. The New endpoint listener wizard is displayed.
4. Use the wizard to create the new endpoint listener configuration by completing the following steps. For more information about the properties that you set with the wizard, see “Endpoint listeners [Settings]” on page 2830. You might want to use the values given in “Example values for endpoint listener configuration” on page 2792.
   a. Select listener name and binding type.
      Endpoint listener name
      Type the name of your choice by which the endpoint listener is known. For example: `wsgwsoaphttp`; `wsgwsoaphttp2`; `SOAPJMSChannel1`; `SOAPJMSChannel2`.
      Binding type
      Select the type of binding that this endpoint listener supports. For a SOAP over HTTP or SOAP over HTTPS endpoint listener, select `SOAP/HTTP` or `SOAP/HTTPS`. For a SOAP over JMS endpoint listener select `SOAP/JMS`.
   b. Optional: Configure JMS settings.
      This panel is only displayed if you selected `SOAP/JMS` in the previous panel.
      You can choose to deploy your endpoint listener application to use an activation specification or a listener port. Listener ports are stabilized. For more information, read the article on stabilized features. Wherever possible, you should deploy your endpoint listener application to use an activation specification. You can use only activation specifications with the default messaging provider; you can use either activation specifications or listener ports with the WebSphere MQ messaging provider.
      Select from the drop-down lists the **listener port**, or the **activation specification** and **queue connection factory** that you have previously configured as described in “Configuring JMS resources for the synchronous SOAP over JMS endpoint listener” on page 2795.
   c. Configure required URLs. Configure web addresses for the application root and the WSDL serving root. You can either select pre-configured addresses based on the known virtual hosts, or create new values.
      URL root
      Select or type the address at which external clients access the endpoint listener endpoint.

The URL root is the context root of the endpoint listener application, and provides the root of the web address that is used to build the endpoint addresses within WSDL files to direct requesters to this endpoint listener.

An HTTP server can be used with a stand-alone application sever. Alternatively, if your endpoint listener is used by external clients to access a cluster providing high availability or workload management, your cluster usually employs a suitably configured HTTP server (or WebSphere proxy server) operating as an IP-sprayer. In either case, if external clients access the endpoint listener through an HTTP server or server cluster, using default port 80, then specify the HTTP server name and no port number. For example (for SOAP over HTTP endpoint listener 1):
`http://www.yourcompany.com/wsgwsoaphttp1`
For a stand-alone application server, your endpoint listener is typically configured for clients to connect directly to an individual application server. If your endpoint listener is used by external clients to access a cluster, you can configure the listener so that clients connect directly to an individual application server within the cluster as shown in the following example, but this might restrict the high availability or workload management capabilities of your cluster. However, if you allow external clients to connect direct to your application server (for example because it is a stand-alone server or in a development or test environment) then specify the application server host name and port number. For example (for SOAP over HTTP endpoint listener 1):
`http://your.server.name:9080/wsgwsoaphttp1`
where the port number (specified as 9080 in this example) matches the **WC_defaultHost** port value for the application server concerned.

WSDL serving HTTP URL root

Type the root of the web address for the WSDL files of the inbound services that are available at this endpoint listener. This address comprises the root of the HTTP address at which external clients access your endpoint listener application, followed by `/sibws`. This represents the URL that is used when publishing the WSDL URL to a UDDI registry. The host and port name you specify for the **WSDL serving HTTP URL root** typically match those you specify for the **URL root**.

If external clients access the endpoint listener through an HTTP server or server cluster, typically by using default port 80, then this URL root includes the HTTP server name and no port number. For example:

`http://www.yourcompany.com/sibws`

However, if you allow external clients to connect direct to your application server (for example in a development or test environment) then this URL root includes the application server host name and port number. For example:

`http://your.server.name:9080/sibws`

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service. To get the location details for a given inbound service WSDL file, publish the WSDL file to a compressed file as described in “Modifying an existing inbound service configuration” on page 2780, then look up the location within the exported WSDL file.

d. Select the service integration buses to which the new endpoint listener should be connected. Only buses of which the application server or cluster is a member are available for selection.

5. Click **Finish**.

Results

If the processing completes successfully, the list of endpoint listeners is updated to include the new endpoint listener. Otherwise, an error message is displayed.

What to do next

You are now ready to select this endpoint listener for use with an inbound service as described in “Making an internally-hosted service available as a web service” on page 2778.

Example values for endpoint listener configuration:

You can configure any number of endpoint listeners with values of your own choosing, including the example values given in this topic.

When you create an endpoint listener configuration, you provide the following configuration details:

Endpoint listener name
The name by which the endpoint listener is known.

URL root
The address at which external clients access the endpoint listener endpoint. The URL root is the context root of the endpoint listener application, and provides the root of the web address that is used to build the endpoint addresses within WSDL files to direct requesters to this endpoint listener.

WSDL serving HTTP URL root
The root of the web address for the WSDL files of the inbound services that are available at this endpoint listener. This address comprises the root of the HTTP address at which external clients access your endpoint listener application, followed by `/sibws`.

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service. To get the location details for a given inbound service WSDL file, publish the WSDL file to a compressed file as described in “Modifying an existing inbound service configuration” on page 2780, then look up the location within the exported WSDL file.

The rest of this topic gives example configuration details (endpoint listener name, endpoint listener URL root and WSDL serving URL root) for four endpoint listeners that you might want to create:

- “SOAP over HTTP endpoint listener 1”
- “SOAP over HTTP endpoint listener 2” on page 2793
- “Synchronous SOAP over Java Message Service (JMS) endpoint listener 1” on page 2793
- “Synchronous SOAP over JMS endpoint listener 2” on page 2794

SOAP over HTTP endpoint listener 1

Endpoint listener name
`SOAPHTTPChannel1`.

URL root
The address at which external clients access the endpoint listener endpoint. If external clients access the endpoint listener through an HTTP server or server cluster, by using default port 80, then specify the HTTP server name and no port number. For example (for this endpoint listener):

http://*www.yourcompany.com*/wsgwsoaphttp1However, if you allow external clients to connect direct to your application server (for example because it is a stand-alone server or in a development or test environment) then specify the application server host name and port number. For example (for this endpoint listener):

http://*your.server.name:9080*/wsgwsoaphttp1

WSDL serving HTTP URL root

The root of the HTTP address at which external clients access your endpoint listener application, followed by /sibws. For example:

http://*www.yourcompany.com*/sibws

or

http://*your.server.name:9080*/sibws

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service.

SOAP over HTTP endpoint listener 2

Endpoint listener name

SOAPHTTPChannel2.

URL root

The address at which external clients access the endpoint listener endpoint. If external clients access the endpoint listener through an HTTP server or server cluster, by using default port 80, then specify the HTTP server name and no port number. For example (for this endpoint listener):

http://*www.yourcompany.com*/wsgwsoaphttp2However, if you allow external clients to connect direct to your application server (for example because it is a stand-alone server or in a development or test environment) then specify the application server host name and port number. For example (for this endpoint listener):

http://*your.server.name:9080*/wsgwsoaphttp2

WSDL serving HTTP URL root

The root of the HTTP address at which external clients access your endpoint listener application, followed by /sibws. For example:

http://*www.yourcompany.com*/sibws

or

http://*your.server.name:9080*/sibws

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service.

Synchronous SOAP over Java Message Service (JMS) endpoint listener 1

Endpoint listener name

SOAPJMSChannel1.

URL root

You specify the properties of the synchronous SOAP over JMS endpoint listener 1 endpoint by using the following syntax:

jms:/*queue_or_topic_indicator*?*property_name=property_value* and so on, separating each property using the "&" character.

For example, if you use the default values for queue destination and queue connection factory when you install the synchronous SOAP over JMS endpoint listeners, then the first part of the end point address is:

```
jms:/queue?destination=jms/SOAPJMSQueue1&connectionFactory=jms/SOAPJMSFactory1
```

For each of the synchronous SOAP over JMS endpoint listeners, here is the full list of properties that you can specify in the endpoint address:

Table 260. Property descriptions. The first column of this table lists the property names, and the second column lists the property description.

Property name	Property description
Destination-related Properties (required)	
connectionFactory	The JNDI name of the queue or topic connection factory.
destination	The JNDI name of the destination queue or topic.
JNDI-related Properties (optional)	
initialContextFactory	The name of the initial context factory to use (this is mapped to the java.naming.factory.initial property).
jndiProviderURL	The JNDI provider web address (this is mapped to the java.naming.provider.url property).
JMS-related Properties (optional)	
deliveryMode	Indicates whether the request message is persistent. The valid values are 1 (non persistent) and 2 (persistent). The default value is 1.
timeToLive	The lifetime (in milliseconds) of the request message. A value of 0 indicates an infinite lifetime.
priority	The JMS priority associated with the request message. Valid values are 0 - 9. The default value is 4.
userid	The user ID that is used to gain access to the connection factory.
password	The password that is used to gain access to the connection factory.

WSDL serving HTTP URL root

The root of the HTTP address at which external clients access your endpoint listener application, followed by /SIBWS.

If external clients access the endpoint listener through an HTTP server or server cluster, typically by using default port 80, then this URL root includes the HTTP server name and no port number. For example:

```
http://www.yourcompany.com/sibws
```

However, if you allow external clients to connect direct to your application server (for example in a development or test environment) then this URL root includes the application server host name and port number. For example:

```
http://your.server.name:9080/sibws
```

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service.

Synchronous SOAP over JMS endpoint listener 2

Endpoint listener name

SOAPJMSChannel2.

URL root

You specify the properties of the synchronous SOAP over JMS endpoint listener 2 endpoint by using the following syntax:

jms:/*queue_or_topic_indicator*?*property_name=property_value* and so on, separating each property using the "&" character.

For example, if you use the default values for queue destination and queue connection factory when you install the synchronous SOAP over JMS endpoint listeners, then the first part of the end point address is:

```
jms:/queue?destination=jms/SOAPJMSQueue2&connectionFactory=jms/SOAPJMSFactory2
```

For the full list of properties that can be specified in the endpoint address for synchronous SOAP over JMS endpoint listener 2, see the list of properties detailed above for Synchronous SOAP over JMS endpoint listener 1.

WSDL serving HTTP URL root

The root of the HTTP address at which external clients access your endpoint listener application, followed by `/SIBWS`.

If external clients access the endpoint listener through an HTTP server or server cluster, typically by using default port 80, then this URL root includes the HTTP server name and no port number. For example:

```
http://www.yourcompany.com/sibws
```

However, if you allow external clients to connect direct to your application server (for example in a development or test environment) then this URL root includes the application server host name and port number. For example:

```
http://your.server.name:9080/sibws
```

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service.

Configuring JMS resources for the synchronous SOAP over JMS endpoint listener:

Configure the synchronous SOAP over Java Message Service (JMS) endpoint listeners to use a JMS provider - either the default messaging provider, or another provider such as the WebSphere MQ messaging provider - to pass SOAP messages over JMS.

Before you begin

If you have not already done so, choose a JMS messaging provider.

About this task

If you are defining a SOAP over JMS endpoint listener, you must first configure the following JMS resources for your JMS provider:

- Service integration bus queue destination (for the default messaging provider)
- JMS queue connection factory
- JMS queue destination
- JMS activation specification or listener port

Note:

Listener ports are stabilized. For more information, read the article on stabilized features. Wherever possible, you should deploy your endpoint listener application to use an activation specification. You can use only activation specifications with the default messaging provider; you can use either activation specifications or listener ports with the WebSphere MQ messaging provider.

Procedure

1. Use the administrative console to create and configure queue connection factories and queue destinations.

For more information about how to do this for your messaging provider, see the related links.

Create a queue connection factory and a queue destination for each endpoint listener that you plan to configure. For example, if you plan to configure both of the SOAP over JMS endpoint listeners that are supplied with WebSphere Application Server, create two connection factories (one for each endpoint listener) and two queues. The JMS resources and JNDI names that the supplied SOAP over JMS endpoint listeners expect by default are provided in the following table. If you use different resources and names in this step, then change the defaults when you subsequently configure the endpoint listener.

Table 261. JMS resources and expected JNDI names. The first column of this table lists the JMS resources, the second column shows the expected default JNDI names for endpoint listener 1, the third column shows the expected default JNDI name for endpoint listener 2, the fourth column shows the expected queue name for endpoint listener 1, and the fifth column shows the expected queue name for endpoint listener 2.

JMS resource	default JNDI name (endpoint listener 1)	default JNDI name (endpoint listener 2)	queue name (endpoint listener 1)	queue name (endpoint listener 2)
JMS queue connection factory	jms/SOAPJMSFactory1	jms/SOAPJMSFactory2	Not required	Not required
JMS queue destination	jms/SOAPJMSQueue1	jms/SOAPJMSQueue2	User defined (for example: SOAPJMSDestQueue1)	User defined (for example: SOAPJMSDestQueue2)

2. Configure the underlying destination for each JMS queue.

   Configure these destinations as described in the documentation for your JMS provider. If you are using the default messaging provider, use the administrative console to add the two new queue names specified in the previous table as destinations for your application server as described in Creating a queue for point-to-point messaging. The identifier for the destination should match that defined by the user as the queue name in the previous table.

3. Configure the deployment details for the application.

   If you are using activation specifications, use the administrative console to create and configure the activation specifications as described in “Configuring an activation specification for the default messaging provider” on page 524 or Creating an activation specification for the WebSphere MQ messaging provider. Create two activation specifications, one for each endpoint listener. The default JMS resources and associated names that the synchronous SOAP over JMS endpoint listeners expect are provided in the following table. However, you can use any JNDI name for the activation specification, provided that the EAR file has the same JNDI reference in the administrative console “Binding enterprise beans to listener port names or activation specification JNDI names” panel. If you use different resources and names in this step, change the defaults when you subsequently configure the endpoint listener. You must also stop then restart the application server.

Table 262. Default JMS resource and associated name expected. The first column of this table lists the JMS resources, the second column shows the expected default JNDI names for endpoint listener 1, the third column shows the expected default JNDI name for endpoint listener 2, the fourth column shows the expected queue name for endpoint listener 1, and the fifth column shows the expected queue name for endpoint listener 2.

JMS resource	default JNDI name (endpoint listener 1)	default JNDI name (endpoint listener 2)	destination JNDI name (endpoint listener 1)	destination JNDI name (endpoint listener 2)
activation specification	eis/SOAPJMSChannel1	eis/SOAPJMSChannel2	jms/SOAPJMSQueue1	jms/SOAPJMSQueue2

If you are using listener ports with any supported JMS provider, use the administrative console to create and configure the listener ports in the message listener service as described in Adding a new listener port. Create two listener ports (one for each endpoint listener). The default JMS resources and associated names that the supplied SOAP over JMS endpoint listeners expect are provided in the following table. If you use different resources and names in this step, then change the defaults when you subsequently configure the endpoint listener.

Table 263. Default JMS resources and expected names. The first column of this table lists the JMS resources, the second column shows the expected default JNDI names for endpoint listener 1, the third column shows the expected default JNDI name for endpoint listener 2, the fourth column shows the expected queue name for endpoint listener 1, and the fifth column shows the expected queue name for endpoint listener 2.

JMS resource	default name (for use with SOAP over JMS endpoint listener 1)	default name (for use with SOAP over JMS endpoint listener 2)
listener port	SOAPJMSPort1	SOAPJMSPort2
connection factory	jms/SOAPJMSFactory1	jms/SOAPJMSFactory2
destination	jms/SOAPJMSQueue1	jms/SOAPJMSQueue2

4. Save your changes to the master configuration.
5. Bind the JMS resources by stopping then restarting the application server.

What to do next

You are now ready to create a new SOAP over JMS endpoint listener configuration.

Modifying an existing endpoint listener configuration:

Modify the additional properties for an endpoint listener that has been configured for use with inbound services.

Before you begin

You cannot use this task to modify the general properties of an existing endpoint listener. The reason for this is that, when you create a new endpoint listener, an associated endpoint listener application is automatically installed that uses the same general property values. Therefore if you subsequently change the general properties, you break the link between the configuration and the underlying application. If you have to change the general properties, you must delete and recreate the endpoint listener configuration.

About this task

To list the endpoint listeners, and to view and modify their configuration details, complete the following steps.

Procedure

1. Start the administrative console.
2. In the navigation pane, click one of the following paths:
   - **Servers -> Server Types -> WebSphere application servers -> *server_name* -> Endpoint listeners**
   - **Servers -> Clusters -> WebSphere application server clusters -> *cluster_name* -> Endpoint listeners**

   A list of endpoint listeners is displayed in an endpoint listener collection form.
3. Click the name of an endpoint listener in the list. The current endpoint listener settings for this endpoint listener are displayed.
4. View the following general properties. If you are modifying an endpoint listener that is supplied with WebSphere Application Server, you might want to use the values given in “Example values for endpoint listener configuration” on page 2792.
   Name View the name by which the endpoint listener is known. If this is your own endpoint listener, rather than one that is supplied with WebSphere Application Server, then this name must match the name given in the endpoint listener application that you have installed (that is, the *display name* of the endpoint module within the endpoint application EAR file).

DescriptionDescription
View the (optional) description of the endpoint listener.

URL root
View the address at which external clients access the endpoint listener endpoint. The URL root is the context root of the endpoint listener application, and provides the root of the web address that is used to build the endpoint addresses within WSDL files to direct requesters to this endpoint listener.

If external clients access the endpoint listener through an HTTP server or server cluster, by using default port 80, then specify the HTTP server name and no port number. For example (for SOAP over HTTP endpoint listener 1):
`http://www.yourcompany.com/wsgwsoaphttp1`
However, if you allow external clients to connect direct to your application server (for example because it is a stand-alone server or in a development or test environment) then specify the application server host name and port number. For example (for SOAP over HTTP endpoint listener 1):
`http://your.server.name:9080/wsgwsoaphttp1`

WSDL serving HTTP URL root
View the root of the web address for the WSDL files of the inbound services that are available at this endpoint listener. This address comprises the root of the HTTP address at which external clients access your endpoint listener application, followed by `/sibws`.

If external clients access the endpoint listener through an HTTP server or server cluster, typically by using default port 80, then this URL root includes the HTTP server name and no port number. For example:

`http://www.yourcompany.com/sibws`

However, if you allow external clients to connect direct to your application server (for example in a development or test environment) then this URL root includes the application server host name and port number. For example:

`http://your.server.name:9080/sibws`

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service. To get the location details for a given inbound service WSDL file, publish the WSDL file to a compressed file as described in “Modifying an existing inbound service configuration” on page 2780, then look up the location within the exported WSDL file.

5. Under the additional properties heading, click **Connection properties**. A list of all the service integration buses that are currently connected to this endpoint listener is displayed in a service integration bus connection properties collection form. Add, amend or delete buses in the list of currently-connected buses. To add a new bus, complete the following steps:
   a. Click **New**. The service integration bus connection properties settings form is displayed.
   b. Under the general properties heading, select an available service integration bus from the drop-down list. The bus is selected and the additional properties for the bus are displayed.

      Note: Under the connection properties for the bus there are **Custom properties**. These custom properties are name and value pairs that you can use to set internal system configuration properties. In each pair, the name is a property key and the value is a string value. You use custom properties to define the manner in which the endpoint listener connects to this bus. Included in this set is property name `com.ibm.ws.sib.webservices.replyDestination`, which defines the reply destination name used by the endpoint listener. Do not modify or remove this property, which is set automatically when the service integration bus is associated with the endpoint listener.
6. Under the additional properties heading, click **Associated application**. Details for the application that handles the requests for this endpoint listener are displayed in an enterprise application settings form.

7. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of service integration buses that are connected to this endpoint listener is updated, and the list of endpoint listeners is redisplayed. Otherwise, an error message is displayed.

What to do next

You are now ready to select this endpoint listener for use with an inbound service as described in “Making an internally-hosted service available as a web service” on page 2778.

Deleting endpoint listener configurations:

Use this task to delete endpoint listener configurations using the administrative console.

Before you begin

You cannot delete an endpoint listener that has inbound ports associated with it. If you try to do this, an error is generated. Before you can delete an endpoint listener, you must either delete each associated inbound service or remove each associated port from the port list for the inbound service.

Decide which method to use to configure these resources. You can delete an endpoint listener configuration by using the administrative console as described in this task, or by using the “deleteSIBWSEndpointListener command” on page 2951.

About this task

To remove one or more endpoint listener configurations, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Servers -> Server Types -> WebSphere application servers -> *server_name* -> Endpoint listeners**. A list of endpoint listeners is displayed in an endpoint listener collection form.
3. Select the check box for every endpoint listener configuration that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of endpoint listeners is updated. Otherwise, an error message is displayed.

Working with JAX-RPC handlers and clients

The Java API for XML-based remote procedure calls (JAX-RPC) provides you with a standard way of developing interoperable and portable web services. You can use JAX-RPC handlers, handler lists and client applications with your service integration bus-enabled web services.

About this task

There are two main elements of JAX-RPC that you can use directly with the service integration bus:

- JAX-RPC handlers and handler lists.
- JAX-RPC client applications.

A JAX-RPC handler is a Java class that performs a range of handling tasks.For example: logging messages, or transforming their contents, or terminating an incoming request. To create a JAX-RPC handler, you can use a tool such as IBM Rational Application Developer. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more ports, so that the handler list can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.

JAX-RPC client applications send and receive web service request and response messages. JAX-RPC client applications that use the IBM JAX-RPC run-time environment can do this in a number of different ways, depending on the bindings in the WSDL document that they are developed against, and the configuration data that is used at run time.

Detailed instructions on how to configure JAX-RPC handlers, handler lists and client applications for use with the service integration bus are provided in the following topics:

Procedure

- "Creating a new JAX-RPC handler configuration."
- "Creating a new JAX-RPC handler list" on page 2804.
- "Sending web service messages directly over the bus from a JAX-RPC client" on page 2807.
- "Implementing JAX-RPC handlers to access SDO messages" on page 2809.

Creating a new JAX-RPC handler configuration:

Create a JAX-RPC handler configuration for use, as part of a handler list, with service integration bus-deployed web services. Handlers monitor messages at ports, and take appropriate action depending upon the sender and content of each message.

Before you begin

This task assumes that you have already created your handler. You can do this by using IBM Rational Application Developer or a similar tool. For more information, see the IBM developerWorks article Support for J2EE Web Services in WebSphere Studio Application Developer V5.1 -- Part 3: JAX-RPC Handlers.

You must also make the handler class available to the server that hosts the port for the service that you want to monitor, as detailed in "Loading JAX-RPC handler classes" on page 2801.

About this task

A Java API for XML-based remote procedure calls (JAX-RPC) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To make WebSphere Application Server aware of your handler, and to make the handler available for inclusion in one or more handler lists, you use the administrative console to create a new handler configuration.

Procedure

1. In the navigation pane, click **Service integration -> Web services -> JAX-RPC Handlers**. The JAX-RPC handlers collection form is displayed.
2. Click **New**. The JAX-RPC handlers settings form is displayed.
3. Type the following general properties:
   Name Type the name by which the handler is known.

   This name must be unique, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.
   - It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

For example `TestHandler`.

Description
Type the (optional) description of the handler.

Class name
Type the name of the class that is to be instantiated. For example `com.ibm.jaxrpc.handler.TestHandler`.

Note: You can configure multiple instances of a handler by creating each instance with a different handler name, and pointing to the same handler class.

4. Click **OK**. The general properties for this item are saved, and the additional properties options are made available.
5. Type the following additional properties:

SOAP roles
Add SOAP actor definitions to the list of SOAP roles in which this handler acts. For more information, see the SOAP specification.

JAX-RPC headers
Add JAX-RPC header definitions (Namespace URI and Local part) to the list of JAX-RPC headers against which this handler operates. JAX-RPC headers are SOAP headers that are processed by a JAX-RPC handler.

Custom properties
Add custom properties (name/value pairs, where the name is a property key and the value is a string value that can be used to set internal system configuration properties).

6. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handlers is updated to include the new handler. Otherwise, an error message is displayed.

What to do next

To use this handler, add it to a handler list as described in Creating a new JAX-RPC handler list or Modifying an existing JAX-RPC handler list.

Loading JAX-RPC handler classes:

A JAX-RPC handler interacts with messages as they pass into and out of the service integration bus, therefore you make the handler class available to the server or cluster that hosts the inbound or outbound port for the service that you want to monitor.

Before you begin

This task assumes that you have already created your handler. You can do this by using IBM Rational Application Developer or a similar tool. For more information, see the IBM developerWorks article Support for J2EE Web Services in WebSphere Studio Application Developer V5.1 -- Part 3: JAX-RPC Handlers.

About this task

Before you can configure your JAX-RPC handler for use with service integration bus-deployed web services, you must make the handler class available. If you want to monitor an inbound port, make the handler class available to the server on which the endpoint listener for that port is located. If you want to monitor an outbound port, make the handler class available to the server on which the outbound port destination is localized.

To make the handler class available to the server that hosts the port that you want to monitor, you create a shared library for the class then add the shared library to the class loader for the server.

Procedure

1. Package the class file for your handler as a JAR file, then copy the JAR file into a convenient directory.

   Make the handler class available to the application server in one of the following ways:

   - Copy the individual class file into a directory structure under *app_server_root*/classes that matches the package name of the class, where *app_server_root* is the root directory for the installation of WebSphere Application Server. For example a handler class com.ibm.jaxrpc.handler.TestHandler is copied into the *app_server_root*/classes/com/ibm/jaxrpc/handler directory.
   - Package the class files for all your handlers as a JAR file, then copy it into the *app_server_root*/lib/app directory.

2. Start the administrative console.
3. Create a shared library for the JAR file.
   a. Navigate to **Environment -> Shared libraries**.
   b. Set the scope at which you want the new library to be visible, then click **New**.
   c. Give the new library a name.
   d. Set the class path to the directory and file name for your handler JAR file.
   e. Save your changes to the master configuration.

   For more information, see Creating shared libraries.
4. Create a class loader for the server on which you want to make the JAR file available.
   a. Navigate to **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server Infrastructure] Java and Process Management -> Class loader**.
   b. Click **New**.
   c. Click **OK**.
   d. Save your changes to the master configuration.

   For more information, see Configuring class loaders of a server.
5. Add the shared library to the class loader for the server.
   a. Navigate to **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server Infrastructure] Java and Process Management -> Class loader -> *class_loader_name* > [Additional Properties] Shared library references**.
   b. Click **Add**.
   c. Click on the name of your new library, then click **OK**.
   d. Save your changes to the master configuration.

   For more information, see Associating shared libraries with servers.

What to do next

You are now ready to configure your handler for use (as part of a handler list) with service integration bus-enabled web services.

Modifying an existing JAX-RPC handler configuration:

A Java API for XML-based remote procedure calls (JAX-RPC) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. Handlers monitor messages at ports, and take appropriate action depending upon the sender and content of each message. Modify the configuration details for a JAX-RPC handler that has been configured for use, as part of a handler list, with service integration bus-deployed Web services.

Before you begin

If you modify a handler class but do not change the class name, you do not have to modify the handler configuration as described in this topic. You just have to stop then restart the servers that host the ports that this handler monitors.

About this task

To list the handlers, and to view and modify their configuration details, complete the following steps.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> JAX-RPC Handlers**. A list of handlers is displayed in a JAX-RPC handlers collection form.
3. Click the name of a handler in the list. The current JAX-RPC handlers settings for this handler are displayed.
4. Modify the following general properties:
   Name Modify the name of the handler.

   This name must be unique, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.
   - It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

   Note: When you change a handler name, the system looks up all objects that refer to it and updates the name.

   Description
   Modify the (optional) description of the handler.

   Class name
   Change the name of the class that is to be instantiated. If you change the class name, you must also make the new class available to the servers that host the ports that this handler monitors, as detailed in "Loading JAX-RPC handler classes" on page 2801.
5. Modify the following additional properties:
   SOAP roles
   Add, modify or remove SOAP actor definitions from the list of SOAP roles in which this handler acts. For more information, see the SOAP specification.

   JAX-RPC headers
   Add, modify or remove JAX-RPC header definitions (Namespace URI and Local part) from the list of JAX-RPC headers against which this handler operates. JAX-RPC headers are SOAP headers that are processed by a JAX-RPC handler.

   Custom properties
   Add, modify or remove custom properties (name/value pairs, where the name is a property key and the value is a string value that can be used to set internal system configuration properties).
6. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handlers is redisplayed. Otherwise, an error message is displayed.

Deleting JAX-RPC handler configurations:

A Java API for XML-based remote procedure calls (JAX-RPC) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. Delete JAX-RPC handlers that are configured for use (as part of a handler list) with service integration bus-deployed web services.

About this task

When you remove a handler that is currently used by one or more web services on a service integration bus, the system removes the handler from the handler lists for each associated web service.

To remove one or more handlers that are currently configured for a service integration bus, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Web services -> JAX-RPC Handlers**. A list of handlers is displayed in a JAX-RPC handlers collection form.
2. Select the check box for every handler that you want to remove.
3. Click **Delete**.

Results

If the processing completes successfully, the list of handlers is updated. Otherwise, an error message is displayed.

Creating a new JAX-RPC handler list:

A Java API for XML-based remote procedure calls (JAX-RPC) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. Create a JAX-RPC handler list for use with service integration bus-enabled web services.

Before you begin

You can only add previously-configured handlers to a handler list. To configure a handler, see Creating a new JAX-RPC handler configuration.

About this task

Handlers monitor messages at ports, and take appropriate action depending upon the sender and content of each message. To enable handlers to undertake more complex operations, you chain them together into handler lists. The approach taken in WebSphere Application Server is to apply handler lists (rather than individual handlers) at the ports, where each handler list contains one or more handlers.

To create a new JAX-RPC handler list, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Web services -> JAX-RPC Handler Lists**. The JAX-RPC handler lists collection form is displayed.
2. Click **New**. The JAX-RPC handler lists settings form is displayed.
3. Type the following general properties:
   Name Type the name by which the handler list is known.

   This name must be unique, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.

- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

For example TestList.

Description

Type the (optional) description of the handler list.

JAX-RPC handlers

In the JAX-RPC handlers pane, complete the following steps:

a. Select one or more handlers from the list of available JAX-RPC handlers, then click **Add** to move the selected handlers into the list of handlers for this JAX-RPC handler list.
b. Select a handler in the list of handlers for this JAX-RPC handler list, then click **Up** or **Down** to change the position of the handler within the list.

Handlers are applied in the sequence in which they appear in the handler list.

Note: If you click **Reset**, only the **Name** and **Description** fields are reset to their state when the form was first loaded. The two lists of available and assigned handlers are not reset.

4. Click **OK**. The general properties for this item are saved.
5. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handler lists is updated to include the new handler list. Otherwise, an error message is displayed.

What to do next

To use this handler list, select it for use with a web service as described in “Modifying an existing inbound service configuration” on page 2780 or “Modifying an existing outbound service configuration” on page 2786.

Modifying an existing JAX-RPC handler list:

A Java API for XML-based remote procedure calls (JAX-RPC) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. Modify the configuration details for a JAX-RPC handler list that has been configured for use with service integration bus-deployed web services.

Before you begin

You can only add previously-configured handlers to a handler list. To configure a handler, see Creating a new JAX-RPC handler configuration.

About this task

Handlers monitor messages at ports, and take appropriate action depending upon the sender and content of each message. To enable handlers to undertake more complex operations, you chain them together into handler lists. The approach taken in WebSphere Application Server is to apply handler lists (rather than individual handlers) at the ports, where each handler list contains one or more handlers.

To list the existing handler lists, and to view and modify their configuration details, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Web services -> JAX-RPC Handler Lists**. A list of all the handler lists is displayed in a JAX-RPC handler lists collection form.

2. Click the name of a handler list in the list. The current JAX-RPC handler lists settings for this handler are displayed.
3. Modify the following general properties:
   Name Modify the name of the handler list.

   This name must be unique, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.
   - It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

   When you change a handler list name, the system looks up all objects that refer to it and updates the name.

   Description
   Modify the (optional) description of the handler list.

   JAX-RPC handlers
   In the JAX-RPC handlers pane, complete the following steps:
   a. Select one or more previously-configured handlers from either the list of available JAX-RPC handlers or the list of handlers for this JAX-RPC handler list, then click **Add** or **Remove** to modify the list of handlers for this JAX-RPC handler list.
   b. Select a handler in the list of handlers for this JAX-RPC handler list, then click **Up** or **Down** to change the position of the handler within the list.

   Handlers are applied in the sequence in which they appear in the handler list.

   Note: If you click **Reset**, only the **Name** and **Description** fields are reset to their state when the form was first loaded. The two lists of available and assigned handlers are not reset.
4. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handler lists is redisplayed. Otherwise, an error message is displayed.

What to do next

To use this handler list, select it for use with a web service as described in "Modifying an existing inbound service configuration" on page 2780 or "Modifying an existing outbound service configuration" on page 2786.

Deleting JAX-RPC handler lists:

A Java API for XML-based remote procedure calls (JAX-RPC) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. Delete JAX-RPC handler lists that are configured for use with service integration bus-deployed Web services.

About this task

When you remove a handler list that is currently used by one or more web services on a service integration bus, the system removes the handler list for each associated web service.

To remove one or more handler lists, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> Web services -> JAX-RPC Handler Lists**. A list of handler lists is displayed in a JAX-RPC handler lists collection form.
2. Select the check box for every handler list that you want to remove.

3. Click **Delete**.

Results

If the processing completes successfully, the list of handler lists is updated. Otherwise, an error message is displayed.

Sending web service messages directly over the bus from a JAX-RPC client:

Use this task to send web service messages over a bus by retargeting the JAX-RPC client.

About this task

Java API for XML-based remote procedure calls (JAX-RPC) client applications send and receive web service request and response messages. JAX-RPC client applications that use the IBM JAX-RPC run-time environment can do this in a number of different ways, depending on the bindings in the WSDL document that they are developed against, and the configuration data that is used at run time.

For an introduction to basic JAX-RPC programming concepts, including the JAX-RPC client and server programming models, see Getting Started with JAX-RPC.

If you want to use a JAX-RPC client to send messages over the service integration bus, you have two choices:

- Use a SOAP binding (SOAP over HTTP or SOAP over JMS), and pass messages indirectly through an endpoint listener to an inbound service. You would do this if you had SOAP-specific JAX-RPC handlers that must run in the client application context.
- Pass messages directly into the service integration bus at a destination by “retargeting” the JAX-RPC client application as described in this topic.

  Note: There are currently limitations regarding the Java types used by services that are retargeted through a JAX-RPC client application.

Retargeting involves setting the following two values into the client application deployment descriptor, or specifying them dynamically at run time from within the client application:

- The *binding namespace* is set to indicate that the client uses the messaging bus directly.
- The *endpoint address* is set to include the particular destination and (optionally) the format of messages that the client uses.

The destination also needs to be configured so that it knows the port type of messages that the JAX-RPC client is using. There are two ways to achieve this:

- Create an outbound service. An outbound service represents an externally-provided web service. In this case, requests from the JAX-RPC client pass through the service destination and are then sent on to the service provider defined by the outbound service configuration.
- Create an inbound service. An inbound service represents a service provided somewhere within or beyond the messaging bus. You can create an inbound service on any existing destination. The creation of an inbound service associates a WSDL port type with the destination. When retargeting to a destination with an inbound service, the client application needs to specify both the destination name and inbound service name, because it is possible to configure more than one inbound service against a single destination. In this case, requests from the JAX-RPC client pass through the destination and then onwards through the service integration bus depending on routing that is done at the initial destination.

To have web service messages sent directly to a destination using a JAX-RPC client, complete the following steps:

Procedure

1. Create the JAX-RPC client application.
2. Create the outbound service or inbound service with which you want the JAX-RPC client application to exchange messages.
3. Use the administrative console to access the port information for your JAX-RPC client application, as described in “Configuring web services client bindings” on page 2605 and Web services client port information.
4. Override the default SOAP binding for your JAX-RPC client application. Change the binding namespace to `http://www.ibm.com/ns/2004/02/wsdl/mp/sib`
5. Override the endpoint that your JAX-RPC client application uses to send web service requests. The new endpoint should use the sib: URL syntax and include either the outbound service destination name, or both the inbound service name and its corresponding destination name.

What to do next

After you change the *binding namespace*, any JAX-RPC handler lists that were configured for the retargeted port are ignored. For clients that are developed against WSDL with a SOAP binding, retargeting directly to the bus causes the handlers to be ignored. However if the client is developed against the non-bound WSDL for the service, retargeting to the bus is not considered to be changing the binding namespace, and so the handler information is retained. In this case the JAX-RPC handlers are called with the SDOMessageContext subclass.

Associated reference information:

- “sib: URL syntax”

sib: URL syntax:

The `sib:` URL has the following syntax:

`sib:/[destination|path]?`*`property_1=value_1`*`&`*`property_2=value_2`*`&...`

where:

- Square brackets (“[ ]”) indicate that a parameter is optional.
- Transport type is `sib:`, followed by either `/destination` to specify destination type or `/path` to specify a forward routing path, followed by a “query string” that contains one or more properties. The permitted properties are described in the subsequent sections of this topic.

Required properties

The following properties are required. They are used to specify the destination for the request.

Note: All destination names must be fully-qualified. That is, they must include the name of the service integration bus as well as the destination name itself. Use the syntax *`bus:destination`*. If a bus or destination name contains a colon or comma, wrap the name in double quotation marks (“”). If it contains a double quotation mark, repeat the quotation mark.

destinationName
The destination name.

path The forward routing path, in the form of a sequence of destination names separated by commas.

replyDestinationName
The name of the destination to be used for the reply.

inboundService
The name of the inbound service that identifies the specific attachment that the requester

application uses. You can omit this value if the destination is a service destination with an associated outbound service configuration, because in that case the requester is attaching to the outbound service through the service destination.

timeout
The time the requester waits for a response. The default value is 60 seconds. A zero value indicates an unlimited wait.

Service integration technologies-related properties

The following properties are optional. If you do not specify a value for a property, then the default value is used. For more information regarding the permitted values for these properties, see the generated API information for the SIMessage interface.

reliability
The reliability of the request message.

timeToLive
The amount of time (in milliseconds) before the request times out. A zero value indicates that the request never times out.

Note: The **timeout** property (see the required properties) is the time delay after which the requester application blocks the application thread that is waiting for a response to a request and response operation. The **time to live** and **replyTimeToLive** optional properties indicate how long the request and reply messages should be processed by the messaging engines. This does not include the processing time at the service implementation. You would therefore usually set the timeout to be the sum of the request and response times to live, plus some amount for the service processing time.

priority
The priority of the request message.

user
The user ID required to access the request destination.

password
The password required to access the request destination.

replyReliability
The reliability of the reply message.

replyTimeToLive
The amount of time (in milliseconds) before the reply times out. A zero value indicates that the reply never times out.

replyPriority
The priority of the reply message.

Other properties

You can also include user-defined properties in the URL. These properties must be named with a `user.` prefix. For example:

```
sib:/destination?destinationName=myBus:myDestination & reliability=assured & user.customData=XYZ
```

After the request is sent, the URL itself is available within the message properties, named `inbound.url`.

Implementing JAX-RPC handlers to access SDO messages:

JAX-RPC handlers are invoked during the processing of request and response messages. For messages that are exchanged by using the SOAP protocol, each JAX-RPC handler is passed a SOAP-specific

MessageContext object. For other protocols, the IBM web services runtime environment passes a MessageContext object that provides a Service Data Objects view of the message. Service Data Objects (SDO) is an open standard for enabling applications to handle data from different data sources in a uniform way, as data graphs.

If the JAX-RPC handler only deals with message context properties, it does not have to be aware of the particular subclass of MessageContext that it is given, because the context property methods are defined by the MessageContext interface itself. If the handler needs to process information contained within the message, it must be coded to work with the required subclasses. Your JAX-RPC handlers should test whether the MessageContext is an instance of the required subclass.

A JAX-RPC handler is given an SDO-specific MessageContext object (an instance of the com.ibm.websphere.webservices.handler.sdo.SDOMessageContext class) rather than the SOAP-specific MessageContext object in the following cases:

- A JAX-RPC client or outbound invocation from the service integration bus invokes a service by using the EJB binding.
- A JAX-RPC client is developed against a non-bound WSDL and is retargeted to a destination in the service integration bus.

The SDOMessageContext class provides methods to get and set the com.ibm.websphere.sdo.SDOMessage instance that represents the message that is being processed. The SDOMessage has a method to access the SDO DataGraph object that holds the message content as SDO DataObjects.

A JAX-RPC handler can modify the SDO DataGraph contents, but it cannot change the format or schema of the message.

The following example shows code that can be used to access the SDO DataGraph from the MessageContext object in a JAX-RPC handler handleRequest method:

```
public boolean handleRequest(MessageContext messageContext) {

  // Convert the MessageContext into an SDOMessageContext
  if( messageContext instanceof SDOMessageContext) {
   SDOMessageContext smc = (SDOMessageContext)messageContext;

   // Retrieve the message
   SDOMessage message = smc.getSDOMessage();

   // Get the root object in the SDO DataGraph
   DataGraph graph = message.getDataGraph();
   DataObject content = graph.getRootObject();

   // Now do something with the message content.....
  }
  return true;
 }
```

Working with mediations

Use a mediation to change the content of a message, or the way in which a message is handled.

Before you begin

For an introduction to using mediations with the service integration bus, see Learning about mediations.

About this task

A mediation is an application that contains a mediation handler. You associate a mediation with a service integration bus destination, and the mediation acts on messages that pass through the destination. The

action taken by a mediation depends upon the specific instructions you give in the mediation handler. For example, you can use a mediation to change the contents of a message, or to choose a particular forward route for a message.

To write a mediation application that contains a mediation handler, install it into WebSphere Application Server and associate it with a bus destination, complete the following steps:

Procedure

1. Create the mediation application. For examples of how to do this, see:
   - Writing a routing mediation
   - Writing a mediation that maps between attachment encoding styles
2. Install the mediation application into WebSphere Application Server.
3. Create the associated mediation object in the bus.

   Note: The **handler list name** is the name you gave to your handler when you defined the handler class as a mediation handler.
4. Mediate the destination. Use the administrative console to access the destination that you want to mediate, then select your mediation from the list of available mediations.

Creating a new UDDI reference

Create a reference (a pointer) to a UDDI registry for use with service integration bus-enabled web services.

Before you begin

This topic assumes that you have already created a J2C authentication alias for each *Authorized Name* that you want to associate with this UDDI reference. For more information about *Authorized Names*, see UDDI registries: Web service directories that can be referenced by bus-enabled web services.

About this task

A UDDI reference is a pointer to a UDDI registry. This registry can be a private UDDI registry such as the IBM WebSphere UDDI Registry, or a public UDDI registry. The service integration technologies interact with UDDI registries in two ways:

- When you configure an inbound service, you can create entries for the web service in one or more UDDI registries.
- When you configure an outbound service, you specify the location of the target WSDL file that describes the web service. This WSDL file can be located at a URL or through a UDDI registry.

In the UDDI model, web services are owned by businesses, and businesses are owned by Authorized Names. Each UDDI reference gives access to the Web services that are owned by a single Authorized Name in a single UDDI registry. For more information about how the service integration technologies work with UDDI registries, see UDDI registries: Web service directories that can be referenced by bus-enabled web services and Publishing a web service to a UDDI registry. For more general information about UDDI and UDDI registries, see the UDDI community at uddi.org.

To create a new UDDI reference, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> UDDI References**. The UDDI references collection form is displayed.
3. Click **New**. The UDDI references settings form is displayed.
4. Type the following general properties:

Name Type the name by which the UDDI reference is known. This name must be unique, and it must follow the following syntax rules:
- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

You might need more than one UDDI reference for a given UDDI registry. For more information, see UDDI registries: Web service directories that can be referenced by bus-enabled web services.

Description
Type the (optional) description of the UDDI reference.

Inquiry URL
Type the web address that provides access to this registry for the SOAP inquiry API.

Publish URL
Type the web address that provides access to this registry for the SOAP publish API.

Authentication alias

Type the J2C authentication alias for the user ID and password of an "Authorized Name" that has update access to this registry.

The alias you enter here must match the user ID of the owner of the corresponding business in the UDDI registry. You can see the owning user ID in UDDI by looking at the business details under the "Authorized Name" field. If the alias you enter here does not match the "Authorized Name" value for the business that owns the service, then the service is not published or found.

If the business has more than one "Authorized Name", you might want to set up multiple UDDI references (each with a different authentication alias) to the same UDDI registry.

5. Click **OK**.

Results

If the processing completes successfully, the list of UDDI references is updated to include the new UDDI reference. Otherwise, an error message is displayed.

What to do next

To use this UDDI reference, select it when "Making an internally-hosted service available as a web service" on page 2778 or "Making an externally-hosted web service available internally" on page 2783.

Modifying an existing UDDI reference:

Modify the configuration details for a UDDI reference that has been configured for use with service integration bus-enabled web services.

About this task

A UDDI reference is a pointer to a UDDI registry. This registry can be a private UDDI registry such as the IBM WebSphere UDDI Registry, or a public UDDI registry. The service integration technologies interact with UDDI registries in two ways:
- When you configure an inbound service, you can create entries for the web service in one or more UDDI registries.
- When you configure an outbound service, you specify the location of the target WSDL file that describes the web service. This WSDL file can be located at a URL or through a UDDI registry.

To list the UDDI references, and to view and modify their configuration details, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> UDDI References**. A list of UDDI references is displayed in a UDDI references collection form.
3. Click the name of a UDDI reference in the list. The current UDDI reference settings for this UDDI reference are displayed.
4. Modify the following general properties:

   Name Modify the name of the UDDI reference.

   This name must be unique, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.
   - It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

   Note: When you change a UDDI reference name, the system looks up all objects that refer to it and updates the name.

   Description
   Modify the (optional) description of the UDDI reference.

   Inquiry URL
   Type the new web address that provides access to this registry for the SOAP inquiry API.

   Publish URL
   Type the new web address that provides access to this registry for the SOAP publish API.

   Authentication alias

   Type the new J2C authentication alias for the user ID and password of an "Authorized Name" that has update access to this registry.

   The alias you enter here must match the user ID of the owner of the corresponding business in the UDDI registry. You can see the owning user ID in UDDI by looking at the business details under the "Authorized Name" field. If the alias you enter here does not match the "Authorized Name" value for the business that owns the service, then the service is not published or found.

   If the business has more than one "Authorized Name", you might want to set up multiple UDDI references (each with a different authentication alias) to the same UDDI registry.
5. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of UDDI references is redisplayed. Otherwise, an error message is displayed.

What to do next

To use this UDDI reference, select it when "Making an internally-hosted service available as a web service" on page 2778 or "Making an externally-hosted web service available internally" on page 2783.

Deleting UDDI references:

delete UDDI references that are configured for use with service integration bus-deployed web services.

About this task

When you remove a UDDI reference that is currently used by one or more web services on a service integration bus, the system removes the UDDI reference for each associated web service and makes any necessary updates in the associated UDDI registry.

To remove UDDI references, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> UDDI References**. A list of UDDI references is displayed in a UDDI references collection form.
3. Select the check box for every UDDI reference that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of UDDI references is updated. Otherwise, an error message is displayed.

Publishing a web service to a UDDI registry:

When you configure an inbound or outbound service, you enable UDDI interaction by associating the service with a UDDI reference, and (depending upon what you are trying to do) either or both of the following pieces of information: The *business key* that identifies the UDDI business category under which you want your service to appear in the UDDI registry, and the service-specific part of the *service key* that the UDDI registry assigns to your service. To help you understand what UDDI business keys and service keys are, and where you find them in a UDDI registry, here is a description of how to publish a web service to a UDDI registry.

About this task

Service integration technologies interact with UDDI registries as described in UDDI registries: Web service directories that can be referenced by bus-enabled web services. When you publish a web service to a UDDI registry, you:

- Specify the type of business that your web service supports. This usually means choosing an existing business type from a list, but you can also create a new business type. For each type of business there is an associated business key. Service integration bus-enabled web services use this key, in combination with the service key, to find the web service in the registry.
- Add a Technical model. Technical models are generic categories. Using these models, a UDDI registry user can search for a type of service, rather than needing to know the access details for a specific service. Bus-enabled web services interact with UDDI registries at the level of individual Web services, and therefore do not use Technical models.
- Add the web service. The UDDI registry assigns a service key to your service, and publishes the service. Bus-enabled web services use this key, in combination with the business key, to find the web service in the registry.

The following steps describe how you publish a web service to the IBM WebSphere UDDI Registry. If you are working with a different UDDI registry, then the specific navigation is different but the underlying principles are the same.

Procedure

1. Specify a business:
   a. To get a list of valid business keys, look up `businesses` in the UDDI registry. Here is an example of a UDDI business key:

      ```
      08A536DC-3482-4E18-BFEC-2E2A23630526
      ```

      .

b. If you do not find an appropriate existing business in the UDDI registry, then use the **Add a business** option on the **Advanced Publish** section of the Publish pane to add a new one.

2. Add a technical model:
   a. Select **Add a technical model** on the **Advanced Publish** section of the Publish pane.
   b. Enter the name as specified for the target namespace of your binding (or interface) WSDL file, then add a description (if required).
   c. Add a category of Type `unspsc` and value `wsdlSpec` (the Key name field can be left blank).
   d. Add an overview URL specifying the web address for your binding WSDL file, then add a description (if required).

      Note: The binding and the service definition for your web service might be held in separate WSDL files, therefore be careful to type the web address of the WSDL file that defines the *binding*.
   e. Click **Publish Technical Model**.
3. Add a service:
   a. Select **Show owned entities** on the **Advanced Publish** section of the Publish pane.
   b. Select **Add a Service** for your business.
   c. Enter the name as specified for the target service in your WSDL file, then add a description (if required).
   d. For the **Access point** verify that the correct web address type is selected (for example `http` for an HTTP access point), then enter the value of the `soap:address` location (or its equivalent) from your service definition WSDL file (for example `http://yourhost:80/SimpleTest/servlet/rpcrouter`).
   e. For the **Technical model** select **Add**, then find the required Technical model by entering a suitable prefix and selecting **Find technical models**, then select the check box for the required Technical model and click **Update**.
   f. Click **Publish Service**.

Results

The UDDI registry assigns a service key to your service, and publishes the service.

What to do next

After the service has been published you can get the service key from the target UDDI registry.

Here is an example of a full UDDI service key:

```
uddi:blade108node01cell:blade108node01:server1:default:6e3d106e-5394-44e3-be17-aca728ac1791
```

The service-specific part of this key is the final part:

```
6e3d106e-5394-44e3-be17-aca728ac1791
```

Creating a new WS-Security binding

Create a new WS-Security binding for use with service integration bus-enabled web services. You use WS-Security bindings to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services.

Before you begin

Use this option to create WS-Security bindings that comply with either the Web Services Security (WS-Security) 1.0 specification, or the previous WS-Security specification, WS-Security Draft 13 (also known as the Web Services Security Core Specification).

Note: Use of WS-Security Draft 13 was deprecated in WebSphere Application Server Version 6.0. Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

This topic assumes that you have got, from the owning parties, the WS-Security bindings for the client (for an inbound service) and the target web service (for an outbound service).

You can only use WS-Security with web service applications that comply with the *Web Services for Java Platform, Enterprise Edition (Java EE)* or *Java Specification Requirements (JSR) 109* specification. For more information, see Web Services Security and Java Platform, Enterprise Edition security relationship. For information about how to make your web service applications JSR-109 compliant, see Implementing JAX-RPC web services clients or Implementing static JAX-WS web services clients .

About this task

WS-Security bindings provide the information that the run-time environment needs to implement the WS-Security configuration (for example “To sign the body, use this key”), You receive this security binding information direct from the service requester or target service provider, in the form of an `ibm-webservicesclient-bnd.xmi` file for the client, and an `ibm-webservices-bnd.xmi` file for the target web service. You extract the information from these `.xmi` files, then manually enter it into the WS-Security bindings forms.

Bindings are administered independently from any web service that uses them, so you can create a binding then apply it to many web services.

WebSphere Application Server also includes a set of default WS-Security binding objects, as described in Default bindings and runtime properties for Web Services Security. However, if you are using either of the single server products WebSphere Application Server or WebSphere Application Server Express, then these default bindings are configured within the application server, and are not available for use with bus-enabled web services.

Unlike most other configuration objects, when you create a WS-Security binding you can only define its basic aspects. To define the binding details you save the new binding, then reopen it for modification as described in Modifying an existing WS-Security binding.

To create a new WS-Security binding, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> WS-Security bindings**. The WS-Security bindings collection form is displayed.
3. Click **New**. The New WS-Security binding wizard is displayed.
4. Use the wizard to assign the following general properties:
   a. Select the version of the WS-Security specification. Set this option to either Draft 13 (for a binding that complies with the WS-Security Draft 13 specification) or 1.0 (for a binding that complies with the Web Services Security (WS-Security) 1.0 specification.

      Note: Use of WS-Security Draft 13 was deprecated in WebSphere Application Server Version 6.0. Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

   b. Specify the binding type.

      Set this option to one of the following binding types:

For WS-Security Version 1.0:

- *request consumer*, for use when consuming requests from a client to an inbound service.
- *request generator*, for use when generating requests from an outbound service to a target web service.
- *response consumer*, for use when consuming responses from a target web service to an outbound service.
- *response generator*, for use when generating responses from an inbound service to a client.

For WS-Security Draft 13:

- *request receiver*, for use when receiving requests from a client to an inbound service.
- *request sender*, for use when sending requests from an outbound service to a target web service.
- *response receiver*, for use when receiving responses from a target web service to an outbound service.
- *response sender*, for use when sending responses from an inbound service to a client.

c. Specify the WS-Security binding.

Give a name to this binding. This name must be unique and it must follow the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

(WS-Security 1.0 bindings only. Optional.) Select the **Use defaults** check box to create a convenient default binding for use in a development and test environment. If you select this option, the binding uses the WebSphere Application Server default set of binding information rather than any custom information that you might subsequently add. Note however that this default binding is by definition insecure, and is not for production use. You can also select or clear this check box when you modify an existing WS-Security binding.

Note: If you are creating a WS-Security 1.0 request generator binding, the web address for the WS-Security 1.0 namespace is displayed in a drop-down list. This is the namespace used by WS-Security 1.0 to send a request, and you should not have to change this value. The other values included in the drop-down list refer to namespaces used by earlier versions of the WS-Security draft specification, and are included for backwards compatibility.

5. Click **Finish**. The general properties for this item are saved.

Results

If the processing completes successfully, the list of WS-Security bindings is updated to include the new binding. Otherwise, an error message is displayed.

What to do next

You are now ready to define the binding details as described in "Modifying an existing WS-Security binding."

Modifying an existing WS-Security binding

You can add or modify the configuration details for a WS-Security binding that is configured for use with service integration bus-enabled web services. You use WS-Security bindings to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services.

About this task

WS-Security bindings provide the information that the run-time environment needs to implement the WS-Security configuration (for example “To sign the body, use this key”). You receive this security binding information direct from the service requester or target service provider, in the form of an `ibm-webservicesclient-bnd.xmi` file for the client, and an `ibm-webservices-bnd.xmi` file for the target web service. You extract the information from these `.xmi` files, then manually enter it into the WS-Security bindings forms.

Bindings are administered independently from any web service that uses them, so you can create a binding then apply it to many web services.

To list the WS-Security bindings, and to view and modify their configuration details, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> WS-Security bindings**. A list of WS-Security bindings is displayed in a WS-Security bindings collection form.

   Each available binding is flagged as one of the following binding types:

   For WS-Security Version 1.0:

   - *request consumer*, for use when consuming requests from a client to an inbound service.
   - *request generator*, for use when generating requests from an outbound service to a target web service.
   - *response consumer*, for use when consuming responses from a target web service to an outbound service.
   - *response generator*, for use when generating responses from an inbound service to a client.

   For WS-Security Draft 13:

   - *request receiver*, for use when receiving requests from a client to an inbound service.
   - *request sender*, for use when sending requests from an outbound service to a target web service.
   - *response receiver*, for use when receiving responses from a target web service to an outbound service.
   - *response sender*, for use when sending responses from an inbound service to a client.

   Each available binding is also flagged as complying with either the Web Services Security (WS-Security) 1.0 specification or the WS-Security Draft 13 specification.

   Note: Use of WS-Security Draft 13 was deprecated in WebSphere Application Server Version 6.0. Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.
3. Click the name of a WS-Security binding in the list. The current settings for this WS-Security binding are displayed.
4. Modify the configuration details for this WS-Security binding. For detailed reference information about each value that you can set, click on the associated link in the following tables:

Table 264. Value references for WS-Security 1.0 bindings. The left hand column of this table lists the value references for theWS-Security 1.0 request consumer, and the right hand column lists the value references for the WS-Security 1.0 request generator.

WS-Security 1.0 request consumer	WS-Security 1.0 request generator
• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Token consumer collection" on page 3397 • "Key information collection" on page 3317 • "Key locator collection" on page 3407 • "Collection certificate store collection" on page 3425 • "Trust anchor collection" on page 3417 • "Web Services Security property collection" on page 3411	• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Token generator collection" on page 3386 • "Key information collection" on page 3317 • "Key locator collection" on page 3407 • "Collection certificate store collection" on page 3425 • Properties

Table 265. Value references for WS-Security 1.0 bindings. The left hand column of this table lists the value references for the WS-Security 1.0 response generator, and the right hand column lists the value references for the WS-Security 1.0 response consumer.

WS-Security 1.0 response generator	WS-Security 1.0 response consumer
• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Token generator collection" on page 3386 • "Key information collection" on page 3317 • "Key locator collection" on page 3407 • "Collection certificate store collection" on page 3425 • "Web Services Security property collection" on page 3411	• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Token consumer collection" on page 3397 • "Key information collection" on page 3317 • "Key locator collection" on page 3407 • "Collection certificate store collection" on page 3425 • "Trust anchor collection" on page 3417 • "Web Services Security property collection" on page 3411

Table 266. Value references for Draft 13 WS-Security bindings. The left hand column of this table lists the value references for the WS-Security Draft 13 request receiver, and the right hand column lists the value references for the WS-Security Draft 13 request receiver.

WS-Security Draft 13 request receiver	WS-Security Draft 13 request sender
• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Trust anchor collection" on page 3417 • "Collection certificate store collection" on page 3425 • "Key locator collection" on page 3407 • "Trusted ID evaluator collection" on page 3436 • "Login mappings collection" on page 3441	• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Key locator collection" on page 3407 • "Login bindings configuration settings" on page 3458

Table 267. Value references for Draft 13 WS-Security bindings. The left hand column of the table lists the value references for the WS-Security Draft 13 response sender, and the right hand column lists the value references for the WS-Security Draft 13 response receiver.

WS-Security Draft 13 response sender	WS-Security Draft 13 response receiver
• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Key locator collection" on page 3407	• "Signing information collection" on page 3298 • "Encryption information collection" on page 3357 • "Trust anchor collection" on page 3417 • "Collection certificate store collection" on page 3425 • "Key locator collection" on page 3407

5. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of WS-Security bindings is redisplayed. Otherwise, an error message is displayed.

Deleting WS-Security bindings

Delete WS-Security bindings that are configured for use with service integration bus-deployed web services.

About this task

When you remove a WS-Security binding that is currently used by one or more web services on a service integration bus, the system removes the WS-Security binding for each associated web service. To remove WS-Security bindings, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> WS-Security bindings**. A list of WS-Security bindings is displayed in a WS-Security bindings collection form.
3. Select the check box for every WS-Security binding that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of WS-Security bindings is updated. Otherwise, an error message is displayed.

Creating a new WS-Security configuration

Create a new WS-Security configuration for use with service integration bus-deployed web services. You use WS-Security configurations to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services.

Before you begin

Use this option to work with WS-Security configurations that comply with either the Web Services Security (WS-Security) 1.0 specification, or the previous WS-Security specification, WS-Security Draft 13 (also known as the Web Services Security Core Specification).

Note: Use of WS-Security Draft 13 was deprecated in WebSphere Application Server Version 6.0. Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

This topic assumes that you have got, from the owning parties, the WS-Security configurations for the client (for an inbound service) and the target web service (for an outbound service).

You can only use WS-Security with web service applications that comply with the *Web Services for Java Platform, Enterprise Edition (Java EE)* or *Java Specification Requirements (JSR) 109* specification. For more information, see Web Services Security and Java Platform, Enterprise Edition security relationship. For information about how to make your web service applications JSR-109 compliant, see Implementing JAX-RPC web services clients or Implementing static JAX-WS web services clients .

About this task

WS-Security configurations specify the level of security that you require (for example “The body must be signed”). This level of security is then implemented through the run-time information contained in a WS-Security binding. You receive the security configuration information direct from the service requester or target service provider, in the form of an `ibm-webservicesclient-ext.xmi` file for the client, and an `ibm-webservices-ext.xmi` file for the target web service, which contain the information about the levels of

security (integrity, confidentiality and identification) that are required. You extract the information from these `.xmi` files, then manually enter it into the WS-Security configuration forms.

Configurations are administered independently from any web service that uses them, so you can create a configuration then apply it to many web services. However, the security requirements for an inbound service (which acts as a target web service) are significantly different to those required for an outbound service (which acts as a client). Consequently, configurations are further divided by service type (inbound or outbound).

Unlike most other configuration objects, when you create a WS-Security configuration you can only define its basic aspects. To define the details you save the new WS-Security configuration, then reopen it for modification as described in “Modifying an existing WS-Security configuration” on page 2822.

To create a new WS-Security configuration, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> WS-Security configurations**. The WS-Security service configurations collection form is displayed.
3. Click **New**. The New WS-Security Service Configuration wizard is displayed.
4. Use the wizard to assign the following general properties:
   a. Select the version of the WS-Security specification. Set this option to either Draft 13 (for a configuration that complies with the WS-Security Draft 13 specification) or 1.0 (for a configuration that complies with the Web Services Security (WS-Security) 1.0 specification.

      Note: Use of WS-Security Draft 13 was deprecated in WebSphere Application Server Version 6.0. Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

   b. Specify the service type. If you are creating a configuration to secure the SOAP messages that pass between a service requester (client) and an inbound service (which acts as a target web service), select Inbound Service. If you are creating a configuration to secure the SOAP messages that pass between an outbound service (which acts as a client) and a target Web service, select Outbound Service.
   c. Specify the WS-Security configuration type.

      Give a name to this configuration. This name must be unique across both WS-Security Version 1.0 and Draft 13 configurations, and it must follow the following syntax rules:

      - It must not start with “.” (a period).
      - It must not start or end with a space.
      - It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

      (Optionally) Specify an Actor URI for this configuration. WS-Security headers within the consumed request message are only processed if they have the specified Actor URI.
5. Click **Finish**. The general properties for this item are saved.

Results

If the processing completes successfully, the list of WS-Security configurations is updated to include the new configuration. Otherwise, an error message is displayed.

What to do next

You are now ready to define the configuration details as described in “Modifying an existing WS-Security configuration” on page 2822.

Modifying an existing WS-Security configuration

You can add or modify the configuration details for a WS-Security configuration that is configured for use with service integration bus-enabled web services. You use WS-Security configurations to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services.

About this task

WS-Security configurations specify the level of security that you require (for example “The body must be signed”). This level of security is then implemented through the run-time information contained in a WS-Security binding. You receive the security configuration information direct from the service requester or target service provider, in the form of an `ibm-webservicesclient-ext.xmi` file for the client, and an `ibm-webservices-ext.xmi` file for the target web service, which contain the information about the levels of security (integrity, confidentiality and identification) that are required. You extract the information from these `.xmi` files, then manually enter it into the WS-Security configuration forms.

Configurations are administered independently from any web service that uses them, so you can create a configuration then apply it to many web services. However, the security requirements for an inbound service (which acts as a target web service) are significantly different to those required for an outbound service (which acts as a client). Consequently, configurations are further divided by service type (inbound or outbound).

To list the WS-Security configurations, and to view and modify their configuration details, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> WS-Security configurations**. A list of WS-Security configurations is displayed in a WS-Security service configurations collection form.

   Each available configuration is flagged as either `Inbound` or `Outbound`. You use an inbound configuration to secure the SOAP messages that pass between a service requester (client) and an inbound service (which acts as a target web service). You use an outbound configuration to secure the SOAP messages that pass between an outbound service (which acts as a client) and a target web service.

   Each available configuration is also flagged as complying with either the Web Services Security (WS-Security) 1.0 specification or the WS-Security Draft 13 specification.

   Note: Use of WS-Security Draft 13 was deprecated in WebSphere Application Server Version 6.0. Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.
3. Click the name of a WS-Security configuration in the list. The current settings for this WS-Security configuration are displayed.
4. Modify the configuration details for this WS-Security configuration. For detailed reference information about each value that you can set, click on the associated link in the following table:

Table 268. Value references for WS-Security configurations. The left hand column of this table lists the value references for the WS-Security 1.0 inbound configuration, and the right hand column lists the value references for the WS-Security 1.0 outbound configuration.

WS-Security 1.0 inbound configuration	WS-Security 1.0 outbound configuration
Request consumer • Required integrity – Message parts – Nonce – Time stamp • Required confidentiality – Message parts – Nonce – Time stamp • Required security token • Caller – Trust method - Properties – Properties • Add time stamp – Properties • Properties **Response generator** • Actor • Integrity – Message parts – Nonce – Time stamp • Confidentiality – Message parts – Nonce – Time stamp • Security Token • Add time stamp – Properties • Properties	**Request generator** • Actor • Integrity – Message parts – Nonce – Time stamp • Confidentiality – Message parts – Nonce – Time stamp • Security Token • Add time stamp – Properties • Properties **Response consumer** • Required integrity – Message parts – Nonce – Time stamp • Required confidentiality – Message parts – Nonce – Time stamp • Required security token • Caller – Trust method - Properties – Properties • Add time stamp – Properties • Properties

Table 269. Value references for WS-Security Draft 13 configurations. The left hand column of this table lists the value references for the WS-Security Draft 13 inbound configuration, and the right hand column lists the value references for the WS-Security Draft 13 outbound configuration.

WS-Security Draft 13 inbound configuration	WS-Security Draft 13 outbound configuration
Request receiver • Required integrity • Required confidentiality • Login configuration – Custom authentication methods • ID assertion • Add received time stamp • Properties **Response sender** • Actor • Integrity • Confidentiality • Add created time stamp • Properties	**Request sender** • Actor • Integrity • Confidentiality • Login configuration • ID assertion • Add created time stamp • Properties **Response receiver** • Required integrity • Required confidentiality • Add received time stamp • Properties

5. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of WS-Security configurations is redisplayed. Otherwise, an error message is displayed.

Deleting WS-Security configurations

Delete WS-Security configurations that are configured for use with service integration bus-enabled web services.

About this task

When you remove a WS-Security configuration that is currently used by one or more web services on a service integration bus, the system removes the WS-Security configuration for each associated web service. To remove WS-Security configurations, complete the following steps:

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> Web services -> WS-Security configurations**. A list of WS-Security configurations is displayed in a WS-Security service configurations collection form.
3. Select the check box for every WS-Security configuration that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of WS-Security configurations is updated. Otherwise, an error message is displayed.

Passing SOAP messages with attachments through the service integration bus

The service integration technologies support web services that use a SOAP binding to pass attachments in a MIME message.

Before you begin

See the restrictions detailed in Limitations in the support for SOAP with attachments.

About this task

The service integration bus supports SOAP messages that contain either old-style attachments (as described in the SOAP Messages with Attachments W3C Note) or attachments that use the Web Services-Interoperability (WS-I) Attachments Profile Version 1.0 (subject to the Limitations in the support for SOAP with attachments).

Attachments are carried through the service integration bus and passed to the inbound or outbound service. The content MIME type of each attachment is preserved. When the external target service for an outbound service is deployed to a Java API for XML-based Remote Procedure Call (JAX-RPC) compliant server, you can access the attachments on the target service by using the `javax.activation.DataHandler` handler.

If the bus receives a message that contains attachments, and the bus subsequently rewrites the message, then the generated message uses the same attachment style as the received message. To transform attachments from one style to another, you can use a mediation to modify the message.

For more information, see the following topics:

Procedure

- “SOAP Messages with Attachments: WSDL examples”
- “Supporting bound attachments: WSDL examples” on page 2826
- “Locating an attachment by using swaref”
- Writing a mediation that maps between attachment encoding styles

Locating an attachment by using swaref

Use this task to locate message attachments by retrieving the message URI and removing `cid:` from the beginning of the retrieved value.

About this task

When a Web Services-Interoperability (WS-I) Attachments Profile Version 1.0 message uses a SOAP with attachments reference (swaref) to refer to an attachment, the swaref might refer to either bound or non-bound attachments, and the swaref might refer to a single attachment several times. To enable you to locate the correct attachment, service integration technologies stores the value of the URI that is encoded in the message within the SDO data graph for the message body.

When storing the value of an element (or attribute) of type swaref in the data graph, service integration technologies stores the complete URI from the message instance. Therefore when you retrieve the URI you remove `cid:` from the beginning of the retrieved value to find the Content ID of the referenced attachment.

Example

The following example shows how to use the value of a swaref element to locate the correct attachment. This example uses the RPC/Literal WSDL and SOAP message from section 4.4 of Web Services-Interoperability (WS-I) Attachments Profile Version 1.0:

```
DataObject infoNode = graph.getRootObject().getDataObject("info");
String contentId = infoNode.getString("body/ClaimDetail/ClaimForm");

// Cut off the "cid:" part of the string
contentId = contentId.substring(4);

// Locate the value of the attachment
DataObject attachmentEntry =
    infoNode.getDataObject("attachments[contentId=" + contentId + "]");
byte[] data = attachmentEntry.getBytes("data");
```

SOAP Messages with Attachments: WSDL examples

Use this task to see an example and explanation of a WSDL file with an attachment.

Example

The following example WSDL illustrates a simple operation that has one attachment called `attch`:

```
<binding name="MyBinding" type="tns:abc" >
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="MyOperation">
    <soap:operation soapAction=""/>
    <input>
      <mime:multipartRelated>
        <mime:part>
          <soap:body parts="part1 part2 ..." use="encoded" namespace="http://mynamespace"
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding"/>
        </mime:part>
        <mime:part>
```

```
          <mime:content part="attch" type="text/html"/>
        </mime:part>
      </mime:multipartRelated>
    </input>
  </operation>
</binding>
```

In this type of WSDL extension:

- There must be a `part` attribute (in this example `attch`) on the input message for the operation (in this example `MyOperation`). There can be other input parts to `MyOperation` that are not attachments.
- In the binding input there must either be a `<soap:body>` tag or a `<mime:multipartRelated>` tag, but not both.
- For MIME messages, the `<soap:body>` tag is inside a `<mime:part>` tag. There must only be one `<mime:part>` tag that contains a `<soap:body>` tag in the binding input and that must not contain a `<mime:content>` tag as well, because a content type of `text/xml` is assumed for the `<soap:body>` tag.
- There can be multiple attachments in a MIME message, each described by a `<mime:part>` tag.
- Each `<mime:part>` tag that does not contain a `<soap:body>` tag contains a `<mime:content>` tag that describes the attachment itself. The `type` attribute inside the `<mime:content>` tag is not checked or used by the service integration bus. It is there to suggest, to the application that uses the service integration bus, what the attachment contains . Multiple `<mime:content>` tags inside a single `<mime:part>` tag means that the back end service expects a single attachment with a type specified by one of the `<mime:content>` tags inside that `<mime:part>` tag.
- The `parts="..."` attribute inside the `<soap:body>` tag is assumed to contain the names of all the SOAP parts in the message, but not the attachment parts. If there are only attachment parts, specify `parts=""` (empty string). If you omit the `parts` attribute altogether, then the service integration bus assumes ALL parts including the attachments - which causes the attachments to appear twice.

In your WSDL you might have defined a schema for the attachment (for instance as a `binary[]`). The service integration technologies silently ignore this mapping and treat the attachment as a Data Handler.

You do not have to mention unreferenced attachments in the WSDL bindings.

Supporting bound attachments: WSDL examples

Use this task to see examples of WSDL fragments with Web Services-Interoperability (WS-I) Attachments Profile Version 1.0 encoding and SOAP Messages with Attachments encoding.

About this task

Web Services-Interoperability (WS-I) Attachments Profile Version 1.0 defines a convention for constructing the Content ID for a bound attachment. This convention encodes the message part name. Consequently, service integration technologies can recognize a bound attachment whether or not the SOAP body contains elements representing that message part. The convention for constructing a Content ID is as follows:

name=uuid@domain

where *name* is the name of the message part that is being encoded, *uuid* is a globally unique identifier, and *domain* is a domain identifier (for example `my.example.com`).

Note: This approach differs from the SOAP Messages with Attachments encoding scheme, which does not define a conventions for the Content ID but does use elements within the SOAP body to indicate that the message part is encoded as an attachment.

In order to distinguish between the cases, service integration technologies assumes that if a message attachment follows the Version 1.0 convention for constructing the Content ID, then it is a Version 1.0 message.

Example

The following WSDL fragment is for a bound attachment, with message instances that follow both styles:

```
<wsdl:binding name="BoundSoapBinding" type="intf:BoundPortType">
    <soap:binding style="rpc"
                  transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="bound">
      <soap:operation soapAction=""/>
      <wsdl:input>
        <mime:multipartRelated>
          <mime:part>
            <soap:body parts="stringIn" namespace="http://bound"
                       use="literal"/>
          </mime:part>
          <mime:part>
            <mime:content part="attachIn" type="text/xml"/>
          </mime:part>
        </mime:multipartRelated>
      </wsdl:input>
```

The following WSDL fragment is for a SOAP instance that uses Version 1.0 encoding. In this fragment, the message body contains no mention of the `attachIn` part, and the Content ID of the attachment identifies the part that is being encoded.

```
--myBoundary
Content-Type: text/xml
Content-Transfer-Encoding: 7bit
Content-Id: <myStartID>

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns0:bound xmlns:ns0="http://bound">
      <stringIn>some string data</stringIn>
    </ns0:bound>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--myBoundary
Content-Type: text/xml
Content-Transfer-Encoding: 7bit
Content-Id: <attachIn=someUUID@some.domain.name>

<someOtherXMLElement/>
--myBoundary--
```

The following WSDL fragment is for a SOAP instance that uses SOAP Messages with Attachments encoding. In this fragment, the message body does contain a reference to the bound attachment, and the Content ID of the attachment is not constrained.

```
--myBoundary
Content-Type: text/xml
Content-Transfer-Encoding: 7bit
Content-Id: <myStartID>

<SOAP-ENV:Envelope
 xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns0:bound xmlns:ns0="http://bound">
      <stringIn>some string data</stringIn>
      <attachIn href="cid:notTheStart"/>
    </ns0:bound>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--myBoundary
```

```
Content-Type: text/xml
Content-Transfer-Encoding: 7bit
Content-Id: <notTheStart>

<someOtherXMLElement/>
--myBoundary--
```

In the previous two cases there is sufficient information in the message to identify the bound attachment, and in both cases service integration technologies places a bound attachment entry in the attachments list, and places the data from the attachment into the body section of the data graph.

Connection Properties [Collection]

Connection properties define the manner in which an endpoint listener connects to the service integration bus.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere application servers -> *server_name* -> Endpoint listeners -> *listener_name* -> Connection Properties.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This panel contains a list of all the service integration buses that are currently connected to this endpoint listener.

Bus name
The name of this property.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Connection Properties [Settings]

Connection properties define the manner in which an endpoint listener connects to the service integration bus.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere application servers -> *server_name* -> Endpoint listeners -> *listener_name* -> Connection Properties -> *connection-property_name*.

Use this panel to connect this endpoint listener to an available service integration bus.

The service properties define the manner in which the endpoint listener connects to the bus. However you do not currently have to use this option to set connection properties, for the following reasons:

- The only property name that is currently supported is `com.ibm.websphere.sib.webservices.replyDestination`, which defines the reply destination name used by the endpoint listener.
- This property is set automatically when the endpoint listener is configured for the service integration bus.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Bus name:

The name of this property.

Required	Yes
Data type	drop-down list

Endpoint listeners [Collection]

An endpoint listener receives requests from service requester applications within a specific application server or cluster.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere application servers -> *server_name* -> Endpoint listeners.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

An endpoint listener is the point (address) at which incoming messages for a web service are received by a service integration bus. The endpoint listener acts as the ultimate receiver of a SOAP message. The resulting messages that pass across the service integration bus are not then SOAP messages, rather just the data and context that resulted from receiving the SOAP message. Each endpoint listener supports a particular binding. The endpoint listeners that are supplied with WebSphere Application Server support SOAP over HTTP and SOAP over JMS bindings.

Before you can use an endpoint listener, you must complete the following steps.

1. Install the endpoint listener application in WebSphere Application Server.
2. Configure the endpoint listener for an application server and (as part of the configuration) connect the endpoint listener to one or more available service integration buses.
3. On a service integration bus to which you have connected the endpoint listener:
   a. Configure an inbound service and (as part of the configuration) connect the service to a new inbound port.
   b. Associate the new inbound port with the newly-deployed endpoint listener.

A deployed endpoint listener is not used until you deploy a web service that uses the endpoint listener.

Name The name of this endpoint listener.

URL root
The root of the URL that should be used to build the endpoint addresses within WSDL documents to direct requesters to this endpoint listener.

Description
An optional description of the endpoint listener.

Buttons

New	Create a new endpoint listener.
Delete	Delete the selected items.

Endpoint listeners [Settings]

An endpoint listener receives requests from service requester applications within a specific application server or cluster.

To view this page in the console, click the following path:

Servers -> Server Types -> WebSphere application servers -> *server_name* -> Endpoint listeners -> *listener_name*.

An endpoint listener is the point (address) at which incoming SOAP messages for a web service are received by a service integration bus. Each endpoint listener supports a particular binding. Use this panel to configure an endpoint listener for an application server and, as part of the configuration process, connect the endpoint listener to one or more available service integration buses.

The endpoint listeners that you can create with WebSphere Application Server support SOAP over HTTP and SOAP over JMS bindings. Example values for these endpoint listeners are given in “Example values for endpoint listener configuration” on page 2792.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of this endpoint listener.

If you are installing your own endpoint listener application, rather than one that is supplied with WebSphere Application Server, then this name must match the name given in the endpoint listener application that you have installed (that is, the *display name* of the endpoint module within the endpoint application EAR file).

Required No
Data type String

Description:

An optional description of the endpoint listener.

Required No
Data type Text area

URL root:

The root of the URL that should be used to build the endpoint addresses within WSDL documents to direct requesters to this endpoint listener.

The address at which external clients access the endpoint listener endpoint. If external clients access the endpoint listener through an HTTP server or server cluster, by using default port 80, then specify the HTTP server name and no port number.For example (for SOAP over HTTP endpoint listener 1):

`http://www.yourcompany.com/wsgwsoaphttp1`

However, if you allow external clients to connect direct to your application server (for example because it is a stand-alone server or in a development or test environment) then specify the application server host name and port number.For example (for SOAP over HTTP endpoint listener 1):

`http://your.server.name:9080/wsgwsoaphttp1`

Required Yes
Data type Text

WSDL serving HTTP URL root:

WSDL serving HTTP URL root

The root of the web address for the WSDL files of the inbound services that are available at this endpoint listener. This address comprises the root of the HTTP address at which external clients access your endpoint listener application, followed by `/sibws`.

If external clients access the endpoint listener through an HTTP server or server cluster, typically by using default port 80, then this URL root includes the HTTP server name and no port number. For example:
`http://www.yourcompany.com/sibws`

However, if you allow external clients to connect direct to your application server (for example in a development or test environment) then this URL root includes the application server host name and port number. For example:
`http://your.server.name:9080/sibws`

Note: The WSDL serving HTTP URL root is only used internally by other components of WebSphere Application Server (notably the IBM UDDI registry). For all other uses, you access the WSDL file through the endpoint listener endpoint for the inbound service.

Required Yes
Data type Text

Additional Properties

Connection Properties

Connection properties define the manner in which an endpoint listener connects to the service integration bus.

The application that handles the requests for this endpoint listener.

JAX-RPC Handler Lists [Collection]

A JAX-RPC handler list defines an ordered list of JAX-RPC handlers to be invoked against requests and responses.

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handler Lists.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more ports, so that the handler list can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.

When you remove a handler list that is currently used by one or more web services on a service integration bus, the system removes the handler list for each associated web service.

Name The name of this JAX-RPC handler list.

Description
An optional description of the JAX-RPC handler list.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

JAX-RPC Handler Lists [Settings]

A JAX-RPC handler list defines an ordered list of JAX-RPC handlers to be invoked against requests and responses.

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handler Lists -> *handler-list_name*.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more ports, so that the handler list can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of this JAX-RPC handler list.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

When you change a handler list name, the system looks up all objects that refer to it and updates the name.

Required Yes
Data type String

Description :

An optional description of the JAX-RPC handler list.

Required No
Data type Text area

JAX-RPC Handlers:

The JAX-RPC handlers in this list.

Handlers are applied in the sequence in which they appear in the handler list.

Required No
Data type Custom

Buttons

Add	Add a selected handler to the handler list.
Remove	Remove a selected handler from the handler list.
Down	Move the selected handler down the handler list.
Up	Move the selected handler up the handler list.

JAX-WS Handler Lists [Collection]

A JAX-WS handler list defines an ordered list of JAX-WS handlers to be invoked against requests and responses.

To view this page in the console, click the following path:

Service integration -> WS-Notification -> JAX-WS Handler Lists.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more JAX-WS based Version 7.0 WS-Notification services or service points, so that the handler list can monitor WS-Notification activity and take appropriate action depending upon the sender and content of each inbound or outbound message.

When you remove a handler list that is currently used by one or more web services on a service integration bus, the system removes the handler list for each associated web service.

Name The name of the JAX-WS handler list

Description
Description of the JAX-WS handler list

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

JAX-WS Handler Lists [Settings]

A JAX-WS handler list defines an ordered list of JAX-WS handlers to be invoked against requests and responses.

To view this page in the console, click the following path:

Service integration -> WS-Notification -> JAX-WS Handler Lists -> *handler-list_name*.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more JAX-WS based Version 7.0 WS-Notification services or service points, so that the handler list can monitor WS-Notification activity and take appropriate action depending upon the sender and content of each inbound or outbound message.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the JAX-WS handler list

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).

- It must not start or end with a space.
- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

When you change a handler list name, the system looks up all objects that refer to it and updates the name.

Required Yes
Data type String

Description:

Description of the JAX-WS handler list

Required No
Data type Text area

JAX-WS Handlers:

The JAX-WS handlers in the handler list

Handlers are applied in the sequence in which they appear in the handler list.

Required No
Data type Custom

Buttons

Add	Add a selected handler to the handler list.
Remove	Remove a selected handler from the handler list.
Down	Move the selected handler down the handler list.
Up	Move the selected handler up the handler list.

JAX-RPC Handlers [Collection]

A JAX-RPC handler provides customization of web service request or response handling.

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handlers.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. For more detailed information about JAX-RPC and JAX-RPC handlers, see the developerWorks article Support for J2EE Web Services in WebSphere Studio Application Developer V5.1 -- Part 3: JAX-RPC Handlers.

When you remove a handler that is currently used by one or more web services on a service integration bus, the system removes the handler from the handler lists for each associated web service.

Name The name of this JAX-RPC handler.

Class name
The name of the class that implements the JAX-RPC handler.

Description
An optional description of the JAX-RPC handler.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

JAX-RPC Handlers [Settings]

A JAX-RPC handler provides customization of web service request or response handling.

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handlers -> *handler_name*.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of this JAX-RPC handler.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

When you change a handler name, the system looks up all objects that refer to it and updates the name.

Required Yes
Data type String

Description:

An optional description of the JAX-RPC handler.

Required No
Data type Text area

Class name:

The name of the class that implements the JAX-RPC handler.

Before you can create a handler configuration, you must make the handler class available to the application server in one of the following ways:

- Copy the individual class file into a directory structure under *app_server_root*/classes that matches the package name of the class, where *app_server_root* is the root directory for the installation of WebSphere Application Server. For example a handler class com.ibm.jaxrpc.handler.TestHandler is copied into the *app_server_root*/classes/com/ibm/jaxrpc/handler directory.
- Package the class files for all your handlers as a JAR file, then copy it into the *app_server_root*/lib/app directory.

Required	Yes
Data type	String

Additional Properties

JAX-RPC Header
Defines the JAX-RPC headers against which this handler operates.

SOAP Roles
Defines the SOAP roles in which this handler acts

Custom properties
Specifies additional custom properties that you can configure for this service.

JAX-WS Handlers [Collection]

A JAX-WS handler provides customization of web service request or response handling.

To view this page in the console, click the following path:

Service integration -> WS-Notification -> JAX-WS Handlers.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request.

When you remove a handler that is currently used by one or more web services on a service integration bus, the system removes the handler from the handler lists for each associated web service.

Name The name of the JAX-WS Handler

Class name
The class name of the JAX-WS Handler

Description
Description of the JAX-WS Handler

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

JAX-WS Handlers [Settings]

A JAX-WS handler provides customization of web service request or response handling.

To view this page in the console, click the following path:

Service integration -> WS-Notification -> JAX-WS Handlers -> *handler_name*.

A handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the JAX-WS Handler

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

When you change a handler name, the system looks up all objects that refer to it and updates the name.

Required Yes
Data type String

Description:

Description of the JAX-WS Handler

Required No
Data type Text area

Class name:

The class name of the JAX-WS Handler

Before you can create a handler configuration, you must make the handler class available to the application server in one of the following ways:

- Copy the individual class file into a directory structure under *app_server_root*`/classes` that matches the package name of the class, where *app_server_root* is the root directory for the installation of

WebSphere Application Server. For example a handler class `com.ibm.jaxws.handler.TestHandler` is copied into the *app_server_root*`/classes/com/ibm/jaxws/handler` directory.

- Package the class files for all your handlers as a JAR file, then copy it into the *app_server_root*`/lib/app` directory.

Required	Yes
Data type	String

JAX-RPC Header [Collection]

Defines the JAX-RPC headers against which this handler operates.

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handlers -> *handler_name* -> JAX-RPC headers.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

JAX-RPC headers are SOAP headers that are processed by a JAX-RPC handler.

Namespace URI
The namespace of the header that is processed by this handler

Local part
The local part of the name of the header that is processed by this handler

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

JAX-RPC Header [Settings]

Defines the JAX-RPC headers against which this handler operates.

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handlers -> *handler_name* -> JAX-RPC headers -> *header_name*.

JAX-RPC headers are SOAP headers that are processed by a JAX-RPC handler.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Namespace URI:

The namespace of the header that is processed by this handler

Required	Yes
Data type	String

Local part:

The local part of the name of the header that is processed by this handler

Required	Yes
Data type	String

Inbound Ports [Collection]

An inbound port describes the web service enablement of a service destination on a specific endpoint listener, with associated configuration.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Inbound Services -> *service_name* -> [Additional Properties] Inbound Ports.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Requests and responses to an inbound service can be sent across any binding (for example SOAP over HTTP or SOAP over JMS) that is available to the service integration bus. Each available binding is represented by a type of port.

Each inbound port is associated with an endpoint listener, and you can control which groups of users can access a particular inbound service by making the service available only through specific endpoint listeners.

You can associate JAX-RPC handler lists with ports, so that the handlers can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port.

You can set the levels of security to be applied to messages. The security level can be set independently for request and response messages.

Note: When a SOAP message is processed at the inbound port as specified in the SOAP specification, if the SOAP header has an actor attribute that is not intended for processing in the inbound port, the attribute is forwarded in the message.

Name The inbound port name. This name appears as the port name within the WSDL published for the inbound service.

Endpoint Listener
An endpoint listener receives requests from service requester applications within a specific application server or cluster.

Description
An optional description of the inbound port. This description appears in any published WSDL for this port.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Inbound Ports [Settings]

An inbound port describes the web service enablement of a service destination on a specific endpoint listener, with associated configuration.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Inbound Services -> *service_name* -> [Additional Properties] Inbound Ports -> *port_name*.

Each inbound port is associated with an endpoint listener, and you can control which groups of users can access a particular inbound service by making the service available only through specific endpoint listeners.

You can set the levels of security to be applied to messages. The security level can be set independently for request and response messages.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The inbound port name. This name appears as the port name within the WSDL published for the inbound service.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required Yes
Data type Text

Description:

An optional description of the inbound port. This description appears in any published WSDL for this port.

Required	No
Data type	Text area

Endpoint listener :

This defines the physical endpoint listener at which requests for this port are received.

Required	Yes
Data type	drop-down list

Template port name:

Name of the port in the template WSDL to use as a basis for this port's binding

Required	No
Data type	drop-down list

JAX-RPC handler list :

This defines which list of JAX-RPC handlers is to be invoked for this port.

You use JAX-RPC handlers to monitor activity at the port, and take appropriate action (for example logging, or re-routing) depending upon the sender and content of each message that passes through the port.

You configure the set of JAX-RPC handler lists by using the administrative console option **Service integration -> Web services -> JAX-RPC Handler Lists**.

Required	No
Data type	drop-down list

Security request binding :

The security binding to be used for requests received by this port.

You configure the security bindings in this list by using the administrative console option **Service integration -> Web services -> WS-Security bindings**.

Required	No
Data type	drop-down list

Security response binding :

The security binding to be used for responses sent by this port.

You configure the security bindings in this list by using the administrative console option **Service integration -> Web services -> WS-Security bindings**.

Required	No
Data type	drop-down list

Security configuration :

Specifies the details of how security is applied to requests and responses.

You define the security configurations in this list by using the administrative console option **Service integration -> Web services -> WS-Security configurations**.

Required	No
Data type	drop-down list

Additional Properties

Custom properties
Specifies additional custom properties that you can configure for this service.

By default, a timeout error is generated if an outbound service waits for more than 60 seconds for a response from a target web service. To change the timeout value for this inbound port from the default value of 60 seconds to a new value of (for example) 120 seconds, create the following custom property:

Name:
timeout

Value (in milliseconds):
120000

Inbound services [Collection]

An inbound service describes the web service enablement of a service destination. It provides the configuration of endpoint listeners within a port.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Inbound Services.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

An inbound service is a web interface to a service that is provided internally (that is, a service provided by your own organization and hosted in a location that is directly available through a service integration bus destination). To configure a locally-hosted service as an inbound service, you first associate it with a service destination, then you configure one or more ports (each with an associated endpoint listener) through which service requests and responses are passed to the service. You can also choose to have the local service made available through one or more UDDI registries.

Name The inbound service name. This name appears as the service name within WSDL published for this inbound service.

Service destination name
The service destination to be enabled for web service access.

Published
Describes the manner in which the WSDL describing an inbound service is published to a UDDI registry.

Description

An optional description of the inbound service. This description appears in any published WSDL for this service.

Buttons

New	New
Delete	Delete the selected items.
Publish to UDDI	Create a new inbound service.
Unpublish from UDDI	Remove the inbound service from the UDDI registry.

Inbound services [Settings]

An inbound service describes the web service enablement of a service destination. It provides the configuration of endpoint listeners within a port.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Inbound Services -> *service_name*.

An inbound service is a web interface to a service that is provided internally (that is, a service provided by your own organization and hosted in a location that is directly available through a service integration bus destination). To configure a locally-hosted service as an inbound service, you first associate it with a service destination, then you configure one or more ports (each with an associated endpoint listener) through which service requests and responses are passed to the service. You can also choose to have the local service made available through one or more UDDI registries.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Buttons

Reload template WSDL	Reload the template WSDL for this inbound service.

General Properties

Name:

The inbound service name. This name appears as the service name within WSDL published for this inbound service.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`
- It must not be longer than 250 characters.

Required Yes
Data type String

Service destination name:

The service destination to be enabled for web service access.

Required	No
Data type	drop-down list

Description:

An optional description of the inbound service. This description appears in any published WSDL for this service.

Required	No
Data type	Text area

Template WSDL location type:

The type of the template WSDL Location.

Specify the location type for the template WSDL file. The template WSDL file is either located at a web address, or through a UDDI registry. When service integration technologies deploys the web service, it use this template file as the basis for generating a WSDL file for the service.

Required	No
Data type	Radio button

Template WSDL location:

The URL location or UDDI service key of the template WSDL.

Depending upon which template WSDL location type you specified in the previous field, enter either the URL location or the service-specific part of the UDDI service key for the template WSDL file.

Here is an example of a full UDDI service key:

```
uddi:blade108node01cell:blade108node01:server1:default:6e3d106e-5394-44e3-be17-aca728ac1791
```

The service-specific part of this key is the final part:

```
6e3d106e-5394-44e3-be17-aca728ac1791
```

Required	Yes
Data type	Text

Template WSDL UDDI registry:

The UDDI registry containing the template WSDL. Required for UDDI location type.

If you specified a template WSDL location type of “UDDI”, select a UDDI reference from the drop-down list.

You configure the UDDI references in this list by using the administrative console option **Service integration -> Web services -> UDDI References**.

Required	No
Data type	drop-down list

Template WSDL service name:

The name of the service within the template WSDL. Only required if there is more than one service in the WSDL.

If the template WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the service name.

Required	Yes
Data type	String

Template WSDL service namespace:

The namespace of the service within the template WSDL. Only required if there is more than one service in the WSDL.

If the template WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the namespace of the service name.

Required	Yes
Data type	String

Enable operation level security:

Indicates whether the access control policy should be enforced.

If you enable this option you must also complete, for this web service, the steps described in the information center for password-protecting a web service operation.

Required	No
Data type	Boolean

Additional Properties

Inbound Ports
An inbound port describes the web service enablement of a service destination on a specific endpoint listener, with associated configuration.

UDDI Publication
Describes the manner in which the WSDL describing an inbound service is published to a UDDI registry.

Custom properties
Specifies additional custom properties that you can configure for this service.

Publish WSDL files to ZIP file
Publish the WSDL files for this service to a .zip file

Outbound Ports [Collection]

An outbound port represents a single port for a WSDL-defined service provider. It provides the configuration of invocation of the web service.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Outbound Services -> *service_name* -> Outbound Ports.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Requests and responses to an outbound service can be sent across any binding (for example SOAP over HTTP or SOAP over JMS) that is available to both the service integration bus and the external web service. Each available binding is represented by a port.

You can associate JAX-RPC handler lists with ports, so that the handlers can monitor activity at the port, and take appropriate action depending upon the sender and content of each message that passes through the port. If the external web service requires HTTP basic authentication, you can use a JAX-RPC handler list to provide an HTTP basic authentication header.

You can set the levels of security to be applied to messages. The security level can be set independently for request and response messages.

Because the service is hosted externally, you might also have to enable proxy server authentication to get access to the Internet.

Name The port name.

Port destination name
The name of the port destination for this port.

Description
An optional description of the port.

Buttons

New	Create a new outbound port.
Delete	Delete the selected items.

Outbound Ports [Settings]

An outbound port represents a single port for a WSDL-defined service provider. It provides the configuration of invocation of the web service.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Outbound Services -> *service_name* -> Outbound Ports -> *port_name*.

Requests and responses to an outbound service can be sent across any binding (for example SOAP over HTTP or SOAP over JMS) that is available to both the service integration bus and the external web service. Each available binding is represented by a port.

You can set the levels of security to be applied to messages. The security level can be set independently for request and response messages.

Because the service is hosted externally, you might also have to enable proxy server authentication to get permission to access the Internet.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The port name.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	Custom

Description:

An optional description of the port.

Required	No
Data type	Text area

Port destination name:

The name of the port destination for this port.

Required	Yes
Data type	Custom

Port destination point:

The application server or cluster for the port destination.

Required	Yes
Data type	Custom

Binding namespace:

The namespace of the binding for this point.

Required	No
Data type	String

Endpoint address:

The endpoint address for this port.

Required No
Data type String

JAX-RPC handler list:

This defines which list of JAX-RPC handlers is to be invoked for this port.

You use JAX-RPC handlers to monitor activity at the port, and take appropriate action (for example logging, or re-routing) depending upon the sender and content of each message that passes through the port. If the external web service requires HTTP basic authentication, you can use a JAX-RPC handler list to provide an HTTP basic authentication header.

You configure the set of JAX-RPC handler lists by using the administrative console option **Service integration -> Web services -> JAX-RPC Handler Lists**.

Required No
Data type drop-down list

Security request binding:

The security binding to be used with requests sent by this port.

You configure the security bindings in this list by using the administrative console option **Service integration -> Web services -> WS-Security bindings**.

Required No
Data type drop-down list

Security response binding:

The security binding to be used for responses received by this port.

You configure the security bindings in this list by using the administrative console option **Service integration -> Web services -> WS-Security bindings**.

Required No
Data type drop-down list

Security configuration:

Specifies the details of how security is applied to requests and responses.

You define the security configurations in this list by using the administrative console option **Service integration -> Web services -> WS-Security configurations**.

Required No
Data type drop-down list

Authenticating proxy host name :

The host name of the authenticating proxy used for invoking requests for this port.

The service integration technologies require access to the Internet for invoking outbound web services and for retrieval of service provider WSDL files. If you use a proxy server in support of Internet routing, and if your proxy server is configured to require authentication before it grants access to the Internet, then you must also complete, for this outbound port, the steps described in the information center for enabling proxy server authentication.

Required	No
Data type	String

Authenticating proxy port number :

The port number of the authenticating proxy.

Required	No
Data type	String

Authenticating proxy Authorization Alias:

The name of the authorization alias that contains the user name and password to use with the authenticating proxy.

Required	No
Data type	String

Additional Properties

Custom properties
Specifies additional custom properties that you can configure for this service.

Outbound services [Collection]

An outbound service represents a WSDL-described service.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Outbound Services.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

An outbound service is a web service that is hosted externally, and is made available through a service integration bus. To configure an externally-hosted service for a bus, you first associate it with a service destination, then you configure one or more port destinations (one for each type of binding, for example SOAP over HTTP or SOAP over JMS) through which service requests and responses are passed to the external service.

Note: You get the port definitions from the WSDL, but you can choose which ones you want to create.

Because the service is hosted externally, you might also have to enable proxy server authentication for each port to get permission to access the Internet.

Name The outbound service name.

Service destination name
The name of the service destination for this outbound service.

Description
Description of the outbound service.

Buttons

New	Create a new outbound service.
Delete	Delete the selected items.

Outbound services [Settings]

An outbound service represents a WSDL-described service.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Outbound Services -> *service_name*.

An outbound service is a web service that is hosted externally, and is made available through a service integration bus. To configure an externally-hosted service for a bus, you first associate it with a service destination, then you configure one or more port destinations (one for each type of binding, for example SOAP over HTTP or SOAP over JMS) through which service requests and responses are passed to the external service.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Buttons

Reload WSDL	Refresh the details displayed with data from the WSDL file.

General Properties

Name:

The outbound service name.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required Yes
Data type String

Description:

Description of the outbound service.

Required No
Data type Text area

Service destination name:

The name of the service destination for this outbound service.

Required No
Data type Custom

WSDL location type:

The web service WSDL location type.

Specify the location type for the service provider WSDL file that describes the web service. The WSDL file is either located at a web address, or through a UDDI registry.

Required No
Data type Radio button

WSDL location:

The URL location or the UDDI service key of the WSDL. When specifying a UDDI service key, provide only the last segment of the key. For example, for a service key of 'uddi:cell01:node01:server1:default:6e3d106e-5394-44e3-be17-aca728ac1791', supply a value of '6e3d106e-5394-44e3-be17-aca728ac1791'.

Depending upon which WSDL location type you specified in the previous field, enter either the URL location or the service-specific part of the UDDI service key for the service provider WSDL file.

Here is an example of a full UDDI service key:

```
uddi:blade108node01cell:blade108node01:server1:default:6e3d106e-5394-44e3-be17-aca728ac1791
```

The service-specific part of this key is the final part:

```
6e3d106e-5394-44e3-be17-aca728ac1791
```

Required Yes
Data type Text

WSDL UDDI Registry:

The UDDI registry containing the service provider's WSDL document. Required for UDDI Location Type.

If you specified a WSDL location type of “UDDI”, select a UDDI reference from the selection list.

You configure the UDDI references in this list by using the administrative console option **Service integration -> Web services -> UDDI References**.

Required No
Data type drop-down list

WSDL service name:

The name of the service within the WSDL. Required if there is more than one service in the WSDL.

If the service provider WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the service name from the service provider WSDL.

Required	Yes
Data type	String

WSDL service namespace:

The namespace of the service within the WSDL. Required if there is more than one service in the WSDL.

If the service provider WSDL contains more than one service, or the WSDL is located through a UDDI registry, type the namespace of the service name from the service provider WSDL.

Required	Yes
Data type	Text

Default port name:

This port is used for all invocations unless it is overridden at runtime.

Requests and responses to this service can be sent across any binding (for example SOAP over HTTP or SOAP over JMS) that is available to both the service integration bus and the external web service. Each available binding is represented by a port. Select the default port that you want the service integration technologies to use for communicating with the external service.

Required	No
Data type	drop-down list

Port selection mediation:

The name of the port selection mediation that may override the default port for each request.

You can use a mediation to override the default port. For example, you can specify that requests from a particular client should always be routed to a particular port whether or not it is the default port.

Required	No
Data type	drop-down list

Bus member:

The bus member to which the port selection mediation is assigned.

Required	No
Data type	drop-down list

Enable operation level security:

Selects whether the access control policy should be enforced.

If you enable this option you must also complete, for this web service, the associated steps described in the information center for enabling operation-level authentication.

Required	No
Data type	Boolean

Additional Properties

Outbound Ports
Each port enabled within this service has its own configuration.

Custom properties
Specifies additional custom properties that you can configure for this service.

Publish WSDL files to ZIP file [Settings]

Publish the WSDL files for this service to a .zip file

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Inbound Services -> *service_name* -> [Additional Properties] Publish WSDL files to ZIP file.

Use the publish WSDL files property to export the template WSDL for this inbound service to a compressed file.

The exported file includes a version of the WSDL file that has no ports (bindings) defined. This non-bound WSDL is intended for use by your colleagues preparing to deploy an inbound service. It gives you a convenient way of sharing information on the planned deployment details for the service among your team. When you finally deploy the inbound service, the associated WSDL must be complete (that is, it must include the binding information). For more information, see the topic “Non-bound WSDL” in the information center.

The non-bound WSDL file is always published in the exported compressed file for the inbound service, along with the bound WSDL file if the inbound service has any ports defined. The compressed file, named *inbound_service_name*.zip, therefore always contains the following files:

- *bus_name*.*inbound_service_name*NonBound.wsdl (this file contains the non-bound service, port and binding for the inbound service).
- *bus_name*.*inbound_service_name*PortTypes.wsdl (this file contains the port type definition for the inbound service).

If the inbound service has one or more ports, then the compressed file additionally contains the following files:

- *bus_name*.*inbound_service_name*Service.wsdl (this file contains the service and port elements for the inbound service).
- *bus_name*.*inbound_service_name*Bindings.wsdl (this file contains the binding elements that correspond to the ports for the inbound service).

If there is an error generating the WSDL then an error page is returned.

SOAP Roles [Collection]

Defines the SOAP roles in which this handler acts

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handlers -> *handler_name* -> SOAP roles.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

For more information, see the SOAP specification.

Role Defines the SOAP roles in which this handler acts

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

SOAP Roles [Settings]

Defines the SOAP roles in which this handler acts

To view this page in the console, click the following path:

Service integration -> Web services -> JAX-RPC Handlers -> *handler_name* -> SOAP roles -> *role_name*.

For more information, see the SOAP specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Role:

Defines the SOAP roles in which this handler acts

Required Yes
Data type String

UDDI Publication [Collection]

Describes the manner in which the WSDL describing an inbound service is published to a UDDI registry.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Inbound Services -> *service_name* -> [Additional Properties] UDDI publication.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

For more general information about UDDI and UDDI registries, see the UDDI community at uddi.org.

Name The name of this UDDI publication property.

UDDI reference
The reference of the UDDI registry to which WSDL is to be published.

Published
Describes the manner in which the WSDL describing an inbound service is published to a UDDI registry.

Description
An optional description of the UDDI publication properties.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.
Publish to UDDI	Create a new inbound service.
Unpublish from UDDI	Remove the inbound service from the UDDI registry.

UDDI Publication [Settings]

Describes the manner in which the WSDL describing an inbound service is published to a UDDI registry.

To view this page in the console, click the following path:

Service integration -> Buses -> *bus_name* -> [Services] Inbound Services -> *service_name* -> [Additional Properties] UDDI publication -> *publication_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of this UDDI publication property.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	String

Description:

An optional description of the UDDI publication properties.

Required	No
Data type	Text area

UDDI reference:

The reference of the UDDI registry to which WSDL is to be published.

A UDDI reference is a pointer to a UDDI registry. The UDDI references in the list are those that you added by using the administrative console option **Service integration -> Web services -> UDDI References**. Select a UDDI reference that can access the UDDI business category under which you want to publish this service.

Required	Yes
Data type	Custom

Business key:

The key of the business within which this service is to be published.

The business key identifies the business category under which you want your service to appear in UDDI. To get a list of valid business keys, look up businesses in a UDDI registry. Here is an example of a UDDI business key: `08A536DC-3482-4E18-BFEC-2E2A23630526`.

Required	Yes
Data type	Custom

Published service key:

The key of the service as published to the UDDI registry.

This is the service-specific part of the UDDI service key.

When a service is published to UDDI, the registry assigns a service key to the service.

After the service has been published you can get the service key from the target UDDI registry.

Here is an example of a full UDDI service key:

```
uddi:blade108node01cell:blade108node01:server1:default:6e3d106e-5394-44e3-be17-aca728ac1791
```

The service-specific part of this key is the final part:

```
6e3d106e-5394-44e3-be17-aca728ac1791
```

Required	No
Data type	String

Custom HTTP URL for WSDL publication:

The HTTP URL root of the servlet that is to serve the WSDL that describes this service.

Required	No
Data type	String

UDDI References [Collection]

A UDDI reference describes the parameters necessary to connect to a particular UDDI registry.

To view this page in the console, click the following path:

Service integration -> Web services -> UDDI References.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A UDDI reference is a pointer to a UDDI registry. This registry can be a private UDDI registry such as the IBM WebSphere UDDI Registry, or a public UDDI registry.

For more general information about UDDI and UDDI registries, see the UDDI community at uddi.org.

Name The name of this UDDI reference.

Description
An optional description of the UDDI Registry.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

UDDI References [Settings]

A UDDI reference describes the parameters necessary to connect to a particular UDDI registry.

To view this page in the console, click the following path:

Service integration -> Web services -> UDDI References -> *UDDI-reference_name*.

A UDDI reference is a pointer to a UDDI registry. This registry can be a private UDDI registry such as the IBM WebSphere UDDI Registry, or a public UDDI registry.

In the UDDI model, web services are owned by businesses, and businesses are owned by Authorized Names. Each UDDI reference gives access to the web services that are owned by a single Authorized Name in a single UDDI registry.

For more general information about UDDI and UDDI registries, see the UDDI community at uddi.org.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of this UDDI reference.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

You might need more than one UDDI reference for a given UDDI registry.

Required	Yes
Data type	String

Description:

An optional description of the UDDI Registry.

Required	No
Data type	Text area

Inquiry URL:

The URL that applications use to inquire on the UDDI registry.

This is the web address that provides access to this registry for the SOAP inquiry API.

Required	Yes
Data type	String

Publish URL:

The URL that applications use to publish to the UDDI registry.

This is the web address that provides access to this registry for the SOAP publish API.

Required	No

Data type String

Authentication Alias:

The user ID for accessing the UDDI repository

This is an authentication alias that you have previously defined for the user ID and password of a UDDI "Authorized Name" that has update access to this registry.

Required No
Data type drop-down list

Actor [Settings]

Defines the Actor URI to be included in the WS-Security headers of a generated message.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Response sender] Actor**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Request sender] Actor**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Actor:

Defines the Actor URI to be included in the WS-Security headers of a generated message.

Required No
Data type String

Add created time stamp [Settings]

Specifies whether a time stamp will be added to any sent message. The time stamp may also contain an expires value.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Response sender] Add created timestamp**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Request sender] Add created timestamp**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Enabled:

When selected, a time stamp will be added to the message.

Required	No
Data type	Boolean

Expires:

The expiration time of the time stamp, defined as an xsd:Duration type.

The expires value is defined as a type of xsd:Duration, and the format must match the following regular expression:

```
-?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+(\\.[0-9]*)?S)?)
```

For example, to specify a timestamp expiration of three minutes, enter PT3M.

Required	No
Data type	String

Add received time stamp [Settings]

Specifies whether a time stamp will be added to any received message.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Request receiver] Add received timestamp**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Response receiver] Add received timestamp**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Enabled:

When selected, a time stamp will be added to the message.

Required	No
Data type	Boolean

Confidentiality [Settings]

Specifies the confidentiality constraints applied to sent messages. Indicates which parts of the message will be encrypted.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Response sender] Confidentiality**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Request sender] Confidentiality**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Body content:

Specifies that the body of the message must be encrypted.

Required	No
Data type	Boolean

Username token:

Specifies that the username token header must be encrypted.

Required	No
Data type	Boolean

Custom authentication methods [Collection]

Specifies custom authentication methods this service will accept.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Request receiver] Login configuration -> [Additional properties] Custom authentication methods.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification).

Name The name of the custom authentication method to accept.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Custom authentication methods [Settings]

Specifies custom authentication methods this service will accept.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Request receiver] Login configuration -> [Additional properties] Custom authentication methods -> *method_name*.

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the custom authentication method to accept.

Required	Yes
Data type	String

ID assertion [Settings]

Specifies the signature method and trust mode to use when ID Assertion is set as an authentication method.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Request receiver] ID assertion**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Request sender] ID assertion**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Signature method:

Specifies the method by which the identity will be asserted.

Required	No
Data type	drop-down list

Trust mode:

Specifies the method by which the identity of the trusted party will be provided.

Required	No
Data type	drop-down list

Inbound WS-Security configuration [Settings]

WS-Security configuration for an inbound request. This defines WS-Security requirements for the request consumed from the client and the response generated. The objects created may be applied to one or more inbound ports.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name*.

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Service type:

The type of service the WS-Security configuration applies to.

Required	No
Data type	String

Name:

The name of the inbound WS-Security configuration.

This name must be unique across both WS-Security Version 1.0 and Draft 13 Inbound configurations, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	String

Actor URI:

WS-Security headers within the consumed request message will only be processed if they have the specified Actor URI.

Required	No
Data type	String

Request receiver

Required integrity
Specifies the integrity constraints received messages must meet. Indicates which parts of the message must be digitally signed.

Required confidentiality
Specifies the confidentiality constraints applied to sent messages. Indicates which parts of the message will be encrypted.

Login configuration
Specifies the authentication methods this service supports. Custom authentication methods can also be defined.

ID assertion
Specifies the signature method and trust mode to use when ID Assertion is set as an authentication method.

Add received time stamp
Specifies whether a time stamp will be added to any received message.

Properties
General properties for the inbound WS-Security configuration.

Response sender

Actor Defines the Actor URI to be included in the WS-Security headers of a generated message.

Integrity
Specifies the integrity constraints applied to sent messages. Indicates which parts of the message will be digitally signed.

Confidentiality
Specifies the confidentiality constraints applied to sent messages. Indicates which parts of the message will be encrypted.

Add created time stamp
Specifies whether a time stamp will be added to any sent message. The time stamp may also contain an expires value.

Properties
General properties for the inbound WS-Security configuration.

Integrity [Settings]

Specifies the integrity constraints applied to sent messages. Indicates which parts of the message will be digitally signed.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Response sender] Integrity**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Request sender] Integrity**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Body:

Specifies that the body of the message must be digitally signed.

Required	No
Data type	Boolean

Time stamp:

Specifies that the time stamp header must be digitally signed.

Required	No
Data type	Boolean

Security token:

Specifies that the security token header must be digitally signed.

Required	No
Data type	Boolean

Login configuration [Settings]

Specifies the authentication methods this service supports. Custom authentication methods can also be defined.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Request receiver] Login configuration.

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification).

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Basic authentication:

Specifies that a username and password is used as an authentication method.

Required	No
Data type	Boolean

Basic authentication Nonce settings:

Specifies Nonce settings for when Basic Authentication is used. Nonce is a randomly generated value.

Required	No
Data type	Custom

ID assertion:

Specifies that ID Assertion is used as an authentication method. An ID Assertion configuration must also be set.

Required	No
Data type	Boolean

Signature:

Specifies that digital signature is used as an authentication method.

Required	No
Data type	Boolean

LTPA:

Specifies that Lightweight Third Party Authentication is used as an authentication method.

Required	No
Data type	Boolean

Additional Properties

Custom authentication methods
Specifies custom authentication methods this service will accept.

Login configuration [Settings]

Specifies the authentication methods this service supports. Custom authentication methods can also be defined.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Request sender] Login configuration.

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification).

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Authentication method:

Specifies the predefined authentication method to use.

Required	Yes
Data type	Custom

Range

None No authentication process is used.

Basic authentication
A username and password is used to authenticate the user.

Use Nonce
Nonce is a randomly-generated, cryptographic token that is used to prevent replay attacks.

Nonce time stamp required
Attaches a time stamp element to the message.

ID assertion
This asserts the authenticated identity of the originating client from a web service to a downstream web service. When identity assertion is used, the authentication decision is performed based only on the name of the identity and not on other information such as passwords.

Signature
This refers to an X.509 certificate that is sent by the client to the server. The certificate is used to authenticate to the user registry that is configured at the server.

Custom authentication method
A custom authentication method can be entered by the user in a text format.

Outbound WS-Security configuration [Settings]

WS-Security configuration for an outbound request. This defines WS-Security requirements for the request generated and response consumed from the target. The objects created may be applied to one or more outbound ports.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name*.

This panel is one of a set of panels that allow you to configure the service integration bus in accordance with WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Service type:

The type of service the WS-Security configuration applies to.

Required	No
Data type	String

Name:

The name of the outbound WS-Security configuration.

This name must be unique across both WS-Security Version 1.0 and Draft 13 Inbound configurations, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	String

Actor URI:

WS-Security headers within the consumed response message will only be processed if they have the specified actor URI.

Required	No
Data type	String

Request sender

Actor Defines the Actor URI to be included in the WS-Security headers of a generated message.

Integrity
Specifies the integrity constraints applied to sent messages. Indicates which parts of the message will be digitally signed.

Confidentiality
Specifies the confidentiality constraints applied to sent messages. Indicates which parts of the message will be encrypted.

Login configuration
Specifies the authentication method to use in the request. May be a predefined or custom authentication method.

ID assertion
Specifies the signature method and trust mode to use when ID Assertion is set as an authentication method.

Add created time stamp
Specifies whether a time stamp will be added to any sent message. The time stamp may also contain an expires value.

Properties
General properties for the outbound WS-Security configuration.

Response receiver

Required integrity
Specifies the integrity constraints received messages must meet. Indicates which parts of the message must be digitally signed.

Required confidentiality
Specifies the confidentiality constraints applied to sent messages. Indicates which parts of the message will be encrypted.

Add received time stamp
Specifies whether a time stamp will be added to any received message.

Properties
General properties for the outbound WS-Security configuration.

Required confidentiality [Settings]

Specifies the confidentiality constraints applied to sent messages. Indicates which parts of the message will be encrypted.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Request receiver] Required confidentiality**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Response receiver] Required confidentiality**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Body content:

Specifies that the body of the message must be encrypted.

Required	No
Data type	Boolean

Username token:

Specifies that the username token header must be encrypted.

Required	No
Data type	Boolean

Required integrity [Settings]

Specifies the integrity constraints received messages must meet. Indicates which parts of the message must be digitally signed.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Request receiver] Required integrity**
- **Service integration -> Web services -> WS-Security configurations -> *draft13-outbound-config_name* -> [Response receiver] Required integrity**

This panel is one of a set of panels that you can use to configure the service integration bus in accordance with the WS-Security Draft 13 specification (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Body:

Specifies that the body of the message must be digitally signed.

Required	No
Data type	Boolean

Time stamp:

Specifies that the time stamp header must be digitally signed.

Required	No
Data type	Boolean

Security token:

Specifies that the security token header must be digitally signed.

Required	No
Data type	Boolean

Actor [Settings]

Defines the Actor URI to be included in the WS-Security headers of a generated message.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Actor.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Actor:

Defines the Actor URI to be included in WS-Security headers of generated request.

Required	No
Data type	String

Actor [Settings]

Defines the Actor URI to be included in WS-Security headers of generated response.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Actor.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Actor:

Defines the Actor URI to be included in WS-Security headers of generated response.

Required	No
Data type	String

Add time stamp [Settings]

When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Add timestamp.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Enabled:

If selected, a time stamp will be added to the message.

Required	No
Data type	Boolean

Dialect:

The expression dialect to use.

Required	No
Data type	drop-down list

Keyword:

Identifies the message part in a way defined by the chosen dialect.

When the `http://www.ibm.com/websphere/webservices/wssecurity/dialect-was` dialect value is selected, the following are valid keyword values:

action Specifies the wsa:Action element.

body Specifies the SOAP body element.

dsigkey
Specifies the key information element, which is used for digital signature.

enckey
Specifies the ds:KeyInfo element, which is used for encryption.

messageid
Specifies the wsa:MessageID element.

relatesto
Specifies the wsa:RelatesTo element.

securitytoken
Specifies any security token elements, for example the wsse:BinarySecurityToken element.

timestamp
Specifies the wsu:Timestamp element. This element determines whether the message is valid based upon the time that the message is sent and then received.

to Specifies the wsa:To element.

When the `http://www.w3.org/TR/1999/REC-xpath-1999116` dialect value is selected, then the keyword value can be any valid XPath expression that points to a part of the message. For example:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']
```

Required	No
Data type	String

Expires:

The expiration time of the time stamp, defined as an xsd:Duration type.

The expires value is defined as a type of xsd:Duration, and the format must match the following regular expression:

```
-?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+(\\.[0-9]*)?S)?)
```

For example, to specify a timestamp expiration of three minutes, enter PT3M.

Required	No
Data type	String

Additional Properties

Properties

Properties associated with the added time stamp.

Add time stamp [Settings]

When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Add timestamp.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Enabled:

If selected, a time stamp will be added to the message.

Required	No
Data type	Boolean

Dialect:

The expression dialect to use.

Required	No
Data type	drop-down list

Keyword:

Identifies the message part in a way defined by the chosen dialect.

When the `http://www.ibm.com/websphere/webservices/wssecurity/dialect-was` dialect value is selected, the following are valid keyword values:

action Specifies the wsa:Action element.

body Specifies the SOAP body element.

dsigkey
Specifies the key information element, which is used for digital signature.

enckey
Specifies the ds:KeyInfo element, which is used for encryption.

messageid
Specifies the wsa:MessageID element.

relatesto
Specifies the wsa:RelatesTo element.

securitytoken
Specifies any security token elements, for example the wsse:BinarySecurityToken element.

timestamp
Specifies the wsu:Timestamp element. This element determines whether the message is valid based upon the time that the message is sent and then received.

to Specifies the wsa:To element.

When the `http://www.w3.org/TR/1999/REC-xpath-1999116` dialect value is selected, then the keyword value can be any valid XPath expression that points to a part of the message. For example:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']
```

Required	No
Data type	String

Expires:

The expiration time of the time stamp, defined as an xsd:Duration type.

The expires value is defined as a type of xsd:Duration, and the format must match the following regular expression:

```
-?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+(\\.[0-9]*)?S)?)
```

For example, to specify a timestamp expiration of three minutes, enter PT3M.

Required	No
Data type	String

Additional Properties

Properties
Properties associated with the added time stamp.

Add time stamp [Settings]

When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Add timestamp.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Enabled:

If selected, a time stamp will be added to the message.

Required	No
Data type	Boolean

Dialect:

The expression dialect to use.

Required	No
Data type	drop-down list

Keyword:

Identifies the message part in a way defined by the chosen dialect.

When the `http://www.ibm.com/websphere/webservices/wssecurity/dialect-was` dialect value is selected, the following are valid keyword values:

action Specifies the wsa:Action element.

body Specifies the SOAP body element.

dsigkey
Specifies the key information element, which is used for digital signature.

enckey
Specifies the ds:KeyInfo element, which is used for encryption.

messageid
Specifies the wsa:MessageID element.

relatesto
Specifies the wsa:RelatesTo element.

securitytoken
Specifies any security token elements, for example the wsse:BinarySecurityToken element.

timestamp
Specifies the wsu:Timestamp element. This element determines whether the message is valid based upon the time that the message is sent and then received.

to Specifies the wsa:To element.

When the `http://www.w3.org/TR/1999/REC-xpath-1999116` dialect value is selected, then the keyword value can be any valid XPath expression that points to a part of the message. For example:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']
```

Required	No

Data type String

Expires:

The expiration time of the time stamp, defined as an xsd:Duration type.

The expires value is defined as a type of xsd:Duration, and the format must match the following regular expression:

```
-?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+(\\.[0-9]*)?S)?)
```

For example, to specify a timestamp expiration of three minutes, enter PT3M.

Required No
Data type String

Additional Properties

Properties
Properties associated with the added time stamp.

Add time stamp [Settings]

When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Add timestamp.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Enabled:

If selected, a time stamp will be added to the message.

Required No
Data type Boolean

Dialect:

The expression dialect to use.

Required No
Data type drop-down list

Keyword:

Identifies the message part in a way defined by the chosen dialect.

When the `http://www.ibm.com/websphere/webservices/wssecurity/dialect-was` dialect value is selected, the following are valid keyword values:

action Specifies the wsa:Action element.

body Specifies the SOAP body element.

dsigkey
Specifies the key information element, which is used for digital signature.

enckey
Specifies the ds:KeyInfo element, which is used for encryption.

messageid
Specifies the wsa:MessageID element.

relatesto
Specifies the wsa:RelatesTo element.

securitytoken
Specifies any security token elements, for example the wsse:BinarySecurityToken element.

timestamp
Specifies the wsu:Timestamp element. This element determines whether the message is valid based upon the time that the message is sent and then received.

to Specifies the wsa:To element.

When the `http://www.w3.org/TR/1999/REC-xpath-1999116` dialect value is selected, then the keyword value can be any valid XPath expression that points to a part of the message. For example:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']
```

Required	No
Data type	String

Expires:

The expiration time of the time stamp, defined as an xsd:Duration type.

The expires value is defined as a type of xsd:Duration, and the format must match the following regular expression:

```
-?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+(\\.[0-9]*)?S)?)
```

For example, to specify a timestamp expiration of three minutes, enter PT3M.

Required	No
Data type	String

Additional Properties

Properties
Properties associated with the added time stamp.

Caller [Collection]

Specifies the security token, signed part or encrypted part used for authentication. If a signed or encrypted part is used, the value of the part attribute must be the name of a defined required integrity or required confidentiality constraint. If a stand-alone security token is used for authentication, then the URI and local name attributes must define the type of security token used for authentication.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Caller.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the caller.

Part Specifies the name of the required integrity or required confidentiality part within the message to be used for authentication.

URI Specifies the namespace URI of the security token to be used for authentication.

Local name
Specifies the local name of the security token to be used for authentication.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Caller [Collection]

Specifies the security token, signed part or encrypted part used for authentication. If a signed or encrypted part is used, the value of the part attribute must be the name of a defined required integrity or required confidentiality constraint. If a stand-alone security token is used for authentication, then the URI and local name attributes must define the type of security token used for authentication.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Caller.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the caller.

Part Specifies the name of the required integrity or required confidentiality part within the message to be used for authentication.

URI Specifies the namespace URI of the security token to be used for authentication.

Local name
Specifies the local name of the security token to be used for authentication.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Caller [Settings]

Specifies the security token, signed part or encrypted part used for authentication. If a signed or encrypted part is used, the value of the part attribute must be the name of a defined required integrity or required confidentiality constraint. If a stand-alone security token is used for authentication, then the URI and local name attributes must define the type of security token used for authentication.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Caller -> *caller_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the caller.

Required Yes
Data type String

Part:

Specifies the name of the required integrity or required confidentiality part within the message to be used for authentication.

Required No
Data type drop-down list

URI:

Specifies the namespace URI of the security token to be used for authentication.

If you specify a Username token or X.509 certificate security token, you do not have to specify a URI. If you specify a custom token, enter the URI of the `QName` for the value type. If you specify Lightweight Third Party Authentication (LTPA), enter the following WebSphere Application Server predefined value type URI: `http://www.ibm.com/websphere/appserver/tokentype/5.0.2`. If you specify Lightweight Third Party Authentication propagation (LTPA_PROPAGATION), enter the following WebSphere Application Server predefined value type URI: `http://www.ibm.com/websphere/appserver/tokentype`.

Required No
Data type String

Local name:

Specifies the local name of the security token to be used for authentication.

WebSphere Application Server has the following predefined **local name** value types:

Username token
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken`

X509 certificate token
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3`

 # X509 certificates in a PKIPath
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1`

A list of X509 certificates and CRLs in a PKCS#7
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7`

LTPA For Lightweight Third Party Authentication, the **local name** value type is LTPA.

LTPA_PROPAGATION
For Lightweight Third Party Authentication token propagation, the **local name** value type is LTPA_PROPAGATION.

Attention:

- If you enter LTPA in the **Local name** field, you must also specify the URI value http://www.ibm.com/websphere/appserver/tokentype/5.0.2 in the **URI** field.
- If you enter LTPA_PROPAGATION in the **Local name** field, you must also specify the URI value http://www.ibm.com/websphere/appserver/tokentype in the **URI** field.
- If you enter any of the other predefined local name value types, you can leave the **URI** field blank. For example, to specify “Username token”, enter http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken in the **Local name** field and do not enter a value in the **URI** field.
- If you specify a custom value type for a custom token, you must specify the local name and the URI of the Quality name (QName) of the value type. For example, you might enter Custom in the **Local name** field, and http://www.ibm.com/custom in the **URI** field.

Required	No
Data type	String

Additional Properties

Trust method
The trust method associated with this caller if IDAssertion is in use for verifying an asserted ID from an intermediary.

Properties
Properties associated with the caller.

Caller [Settings]

Specifies the security token, signed part or encrypted part used for authentication. If a signed or encrypted part is used, the value of the part attribute must be the name of a defined required integrity or required confidentiality constraint. If a stand-alone security token is used for authentication, then the URI and local name attributes must define the type of security token used for authentication.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Caller > ***caller_name***.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the caller.

Required	Yes
Data type	String

Part:

Specifies the name of the required integrity or required confidentiality part within the message to be used for authentication.

Required	No
Data type	drop-down list

URI:

Specifies the namespace URI of the security token to be used for authentication.

Required	No
Data type	String

Local name:

Specifies the local name of the security token to be used for authentication.

Required	No
Data type	String

Additional Properties

Trust method
The trust method associated with this caller if IDAssertion is in use for verifying an asserted ID from an intermediary.

Properties
Properties associated with the caller.

Confidentiality [Collection]

Specifies the confidentiality constraints applied to generated messages. This includes specifying which message parts within the generated message must be encrypted, and the message parts to attach encrypted Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Confidentiality.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the confidentiality constraint.

Order Specifies the processing order of this confidentiality element.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Confidentiality [Collection]

Specifies the confidentiality constraints applied to generated messages. This includes specifying which message parts within the generated message must be encrypted, and the message parts to attach encrypted Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Confidentiality.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the confidentiality constraint.

Order Specifies the processing order of this confidentiality element.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Confidentiality [Settings]

Specifies the confidentiality constraints applied to generated messages. This includes specifying which message parts within the generated message must be encrypted, and the message parts to attach encrypted Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Confidentiality -> *confidentiality_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the confidentiality constraint.

Required	Yes
Data type	String

Order:

Specifies the processing order of this confidentiality element.

Required	Yes
Data type	Integer
Range	0 through 9

Additional Properties

Message parts
Specifies parts of the message affected by this confidentiality constraint.

Nonce
Specifies the encrypted Nonce elements which must be inserted into the generated message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the encrypted time stamp elements which must be inserted in the generated message, and what parts of the message they must be attached to.

Confidentiality [Settings]

Specifies the confidentiality constraints applied to generated messages. This includes specifying which message parts within the generated message must be encrypted, and the message parts to attach encrypted Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Confidentiality -> *confidentiality_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the confidentiality constraint.

Required	Yes
Data type	String

Order:

Specifies the processing order of this confidentiality element.

Required	Yes
Data type	Integer
Range	0 through 9

Additional Properties

Message parts
Specifies parts of the message affected by this confidentiality constraint.

Nonce
Specifies the encrypted Nonce elements which must be inserted into the generated message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the encrypted time stamp elements which must be inserted in the generated message, and what parts of the message they must be attached to.

Inbound WS-Security configuration [Settings]

WS-Security configuration for an inbound request. This defines WS-Security requirements for the request consumed from the client and the response generated. The objects created may be applied to one or more inbound ports.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name*.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

Alternatively, you can configure the bus in accordance with the previous WS-Security specification, WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

You use an inbound configuration to secure the SOAP messages that pass between a service requester (client) and an inbound service (which acts as a target web service). The configuration specifies the level of security that you require (for example “The body must be signed”). This level of security is then implemented through the run-time information contained in the following types of WS-Security binding:

For WS-Security Version 1.0:

- *request consumer*, for use when consuming requests from a client to an inbound service.
- *response generator*, for use when generating responses from an inbound service to a client.

For WS-Security Draft 13:
- *request receiver*, for use when receiving requests from a client to an inbound service.
- *response sender*, for use when sending responses from an inbound service to a client.

WS-Security configurations are administered independently from any web service that uses them, so you can create an inbound configuration then apply it to many inbound services.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Service type:

The type of service the WS-Security configuration applies to.

Required	No
Data type	String

Name:

The name of the inbound WS-Security configuration.

This name must be unique across both WS-Security Version 1.0 and Draft 13 Inbound configurations, and it must obey the following syntax rules:
- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	String

Actor URI:

WS-Security headers within the consumed request message will only be processed if they have the specified Actor URI.

Required	No
Data type	String

Request consumer

Required integrity
Specifies the integrity constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be digitally signed, and the message parts to which attached digitally signed Nonce and time stamp elements are expected.

Required confidentiality
Specifies the confidentiality constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be encrypted, and the message parts to which attached encrypted Nonce and time stamp elements are expected.

Required security token
Specifies accepted stand-alone security tokens within a consumed message. Stand-alone security tokens are those not already used for signature or encryption. Defining a required security token means that messages containing a token of that type will be processed according to the usage assertion. The security token will not be used for authentication unless it is also specified within a caller.

Caller
Specifies the security token, signed part or encrypted part used for authentication. If a signed or encrypted part is used, the value of the part attribute must be the name of a defined required integrity or required confidentiality constraint. If a stand-alone security token is used for authentication, then the URI and local name attributes must define the type of security token used for authentication.

Add time stamp
When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

Properties
General properties for the inbound WS-Security configuration.

Response generator

Actor Defines the Actor URI to be included in WS-Security headers of generated response.

Integrity
Specifies the integrity constraints applied to generated messages. This includes specifying which message parts within the generated message must be digitally signed, and the message parts to attach digitally signed Nonce and time stamp elements to.

Confidentiality
Specifies the confidentiality constraints applied to generated messages. This includes specifying which message parts within the generated message must be encrypted, and the message parts to attach encrypted Nonce and time stamp elements to.

Security Token
Specifies stand-alone security tokens to insert into the generated message. Stand-alone security tokens are those not already used for signature or encryption. Standard and custom security tokens may be defined by URI and local name.

Add time stamp
When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

Properties
General properties for the inbound WS-Security configuration.

Integrity [Collection]

Specifies the integrity constraints applied to generated messages. This includes specifying which message parts within the generated message must be digitally signed, and the message parts to attach digitally signed Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Integrity.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the integrity constraint.

Order Specifies the processing order of this integrity constraint.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Integrity [Collection]

Specifies the integrity constraints applied to generated messages. This includes specifying which message parts within the generated message must be digitally signed, and the message parts to attach digitally signed Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Integrity.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the integrity constraint.

Order Specifies the processing order of this integrity constraint.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Integrity [Settings]

Specifies the integrity constraints applied to generated messages. This includes specifying which message parts within the generated message must be digitally signed, and the message parts to attach digitally signed Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Integrity -> *integrity_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the integrity constraint.

Required	Yes
Data type	String

Order:

Specifies the processing order of this integrity constraint.

Required	Yes
Data type	Integer
Range	0 through 9

Additional Properties

Message parts
Specifies parts of the message affected by this integrity constraint.

Nonce
Specifies the digitally signed Nonce elements which must be inserted into the generated message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the digitally signed time stamp elements which must be inserted in the generated message, and what parts of the message they must be attached to.

Integrity [Settings]

Specifies the integrity constraints applied to generated messages. This includes specifying which message parts within the generated message must be digitally signed, and the message parts to attach digitally signed Nonce and time stamp elements to.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Integrity -> *integrity_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the integrity constraint.

Required	Yes
Data type	String

Order:

Specifies the processing order of this integrity constraint.

Required	Yes
Data type	Integer
Range	0 through 9

Additional Properties

Message parts
Specifies parts of the message affected by this integrity constraint.

Nonce
Specifies the digitally signed Nonce elements which must be inserted into the generated message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the digitally signed time stamp elements which must be inserted in the generated message, and what parts of the message they must be attached to.

Message parts [Collection]

Identifies a specific message part according to the specified dialect and keyword.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity -> *required-integrity_name* -> [Additional Properties] Message parts**
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity -> *integrity_name* -> [Additional Properties] Message parts**
- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality -> *required-confidentiality_name* -> [Additional Properties] Message parts**

- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality -> *confidentiality_name* -> [Additional Properties] Message parts**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the message part definition.

Dialect
The expression dialect to use.

Keyword
Identifies the message part in a way defined by the chosen dialect.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Message Parts [Settings]

Identifies a specific message part according to the specified dialect and keyword.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity -> *required-integrity_name* -> [Additional Properties] Message parts -> *message_part_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity -> *integrity_name* -> [Additional Properties] Message parts -> *message_part_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality -> *required-confidentiality_name* -> [Additional Properties] Message parts -> *message_part_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality -> *confidentiality_name* -> [Additional Properties] Message parts -> *message_part_name***

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the message part definition.

Required Yes

Data type String

Dialect:

The expression dialect to use.

Required Yes
Data type drop-down list

Keyword:

Identifies the message part in a way defined by the chosen dialect.

When the `http://www.ibm.com/websphere/webservices/wssecurity/dialect-was` dialect value is selected, the following are valid keyword values:

Required Integrity

action Specifies the wsa:Action element.

body Specifies the SOAP body element.

dsigkey
Specifies the key information element, which is used for digital signature.

enckey
Specifies the ds:KeyInfo element, which is used for encryption.

messageid
Specifies the wsa:MessageID element.

relatesto
Specifies the wsa:RelatesTo element.

securitytoken
Specifies any security token elements, for example the wsse:BinarySecurityToken element.

timestamp
Specifies the wsu:Timestamp element. This element determines whether the message is valid based upon the time that the message is sent and then received.

to Specifies the wsa:To element.

wsaall Specifies all of the WS-Addressing elements in the SOAP header.

wsafaultto
Specifies the `wsa:FaultTo` WS-Addressing element in the SOAP header.

wsafrom
Specifies the `wsa:From` WS-Addressing element in the SOAP header.

wsareplyto
Specifies the `wsa:ReplyTo` WS-Addressing element in the SOAP header.

wscontext
Specifies the WS-Context header for the SOAP header. For more information, see Propagating work area context over Web services.

Required Confidentiality

bodycontent
Specifies the SOAP body

digestvalue
Specifies the ds:DigestValue element within the ds:Signature element

signature
Specifies an entire signature. You can encrypt the signature element, ds:Signature, by selecting this message part.

Note: If the value of a ds:DigestValue element in a signature needs to be encrypted, the entire parent ds:Signature element must be encrypted. You can use the signature keyword to perform the encryption.

usernametoken
Specifies the wsse:UsernameToken element

When the http://www.w3.org/TR/1999/REC-xpath-1999116 dialect value is selected, then the keyword value can be any valid XPath expression that points to a part of the message. For example:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']
```

Note: Do not use this transform algorithm if you want your configured application to be compliant with the Basic Security Profile (BSP). Instead use http://www.w3.org/2002/06/xmldsig-filter2 to ensure compliance.

Required	Yes
Data type	String

Nonce [Collection]

Attaches a Nonce element to the message part specified by the dialect and keyword attributes. Nonce is a randomly generated value.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity -> *required-integrity_name* -> [Additional Properties] Nonce**
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity -> *integrity_name* -> [Additional Properties] Nonce**
- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality -> *required-confidentiality_name* -> [Additional Properties] Nonce**
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality -> *confidentiality_name* -> [Additional Properties] Nonce**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the Nonce element.

Dialect
The expression dialect to use.

Keyword
The message part to attach the Nonce element to, specified in a way defined by the chosen dialect.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Nonce [Settings]

Attaches a Nonce element to the message part specified by the dialect and keyword attributes. Nonce is a randomly generated value.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity -> *required-integrity_name* -> [Additional Properties] Nonce -> *nonce_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity -> *integrity_name* -> [Additional Properties] Nonce -> *nonce_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality -> *required-confidentiality_name* -> [Additional Properties] Nonce -> *nonce_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality -> *confidentiality_name* -> [Additional Properties] Nonce -> *nonce_name***

When a Nonce is added to the specific parts of a message, it might prevent theft and replay attacks because a generated Nonce is unique. For example, without a Nonce, when a user name token is passed from one machine to another machine by using a non-secure transport, such as HTTP, the token might be intercepted and used in a replay attack. The user name token can be stolen even if you use XML digital signature and XML encryption. However, it might be prevented by adding a Nonce.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the Nonce element.

Required Yes
Data type String

Dialect:

The expression dialect to use.

Required Yes

Data type drop-down list

Keyword:

The message part to attach the Nonce element to, specified in a way defined by the chosen dialect.

When the `http://www.ibm.com/websphere/webservices/wssecurity/dialect-was` dialect value is selected, the following are valid keyword values:

action Specifies the wsa:Action element.

body Specifies the SOAP body element.

dsigkey
Specifies the key information element, which is used for digital signature.

enckey
Specifies the ds:KeyInfo element, which is used for encryption.

messageid
Specifies the wsa:MessageID element.

relatesto
Specifies the wsa:RelatesTo element.

securitytoken
Specifies any security token elements, for example the wsse:BinarySecurityToken element.

timestamp
Specifies the wsu:Timestamp element. This element determines whether the message is valid based upon the time that the message is sent and then received.

to Specifies the wsa:To element.

When the `http://www.w3.org/TR/1999/REC-xpath-1999116` dialect value is selected, then the keyword value can be any valid XPath expression that points to a part of the message. For example:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']
```

Required Yes
Data type String

Outbound WS-Security configuration [Settings]

WS-Security configuration for an outbound request. This defines WS-Security requirements for the request generated and response consumed from the target. The objects created may be applied to one or more outbound ports.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name*.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

Alternatively, you can configure the bus in accordance with the previous WS-Security specification, WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the

WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

You use an outbound configuration to secure the SOAP messages that pass between an outbound service (which acts as a client) and a target web service. The configuration specifies the level of security that you require (for example “The body must be signed”). This level of security is then implemented through the run-time information contained in the following types of WS-Security binding:

For WS-Security Version 1.0:

- *request generator*, for use when generating requests from an outbound service to a target web service.
- *response consumer*, for use when consuming responses from a target web service to an outbound service.

For WS-Security Draft 13:

- *request sender*, for use when sending requests from an outbound service to a target web service.
- *response receiver*, for use when receiving responses from a target web service to an outbound service.

WS-Security configurations are administered independently from any web service that uses them, so you can create an outbound configuration then apply it to many outbound services.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Service type:

The type of service the WS-Security configuration applies to.

Required	No
Data type	String

Name:

The name of the outbound WS-Security configuration.

This name must be unique across both WS-Security Version 1.0 and Draft 13 Inbound configurations, and it must obey the following syntax rules:

- It must not start with “.” (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	String

Actor URI:

WS-Security headers within the consumed response message will only be processed if they have the specified actor URI.

Required	No
Data type	String

Request generator

Actor Defines the Actor URI to be included in the WS-Security headers of a generated message.

Integrity
Specifies the integrity constraints applied to generated messages. This includes specifying which message parts within the generated message must be digitally signed, and the message parts to attach digitally signed Nonce and time stamp elements to.

Confidentiality
Specifies the confidentiality constraints applied to generated messages. This includes specifying which message parts within the generated message must be encrypted, and the message parts to attach encrypted Nonce and time stamp elements to.

Security Token
Specifies stand-alone security tokens to insert into the generated message. Stand-alone security tokens are those not already used for signature or encryption. Standard and custom security tokens may be defined by URI and local name.

Add time stamp
When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

Properties
General properties for the outbound WS-Security configuration.

Response consumer

Required integrity
Specifies the integrity constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be digitally signed, and the message parts to which attached digitally signed Nonce and time stamp elements are expected.

Required confidentiality
Specifies the confidentiality constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be encrypted, and the message parts to which attached encrypted Nonce and time stamp elements are expected.

Required security token
Specifies accepted stand-alone security tokens within a consumed message. Stand-alone security tokens are those not already used for signature or encryption. Defining a required security token means that messages containing a token of that type will be processed according to the usage assertion. The security token will not be used for authentication unless it is also specified within a caller.

Caller
Specifies the security token, signed part or encrypted part used for authentication. If a signed or encrypted part is used, the value of the part attribute must be the name of a defined required integrity or required confidentiality constraint. If a stand-alone security token is used for authentication, then the URI and local name attributes must define the type of security token used for authentication.

Add time stamp
When add time stamp is specified for a consumer, a time stamp is added indicating when the message was consumed. For a generator, a time stamp is added indicating when the message was generated.

Properties
General properties for the outbound WS-Security configuration.

Property collection

Custom properties for the type of service integration resource. Type the name and value of any custom properties that you need.

To view this pane in the console, click one of several paths; for example **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Response sender] Properties**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Name The name of the custom property.

Value The value of the custom property.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Property settings

Custom properties for the type of service integration resource. Type the name and value of any custom properties that you need.

To view this pane in the console, click one of several paths; for example **Service integration -> Web services -> WS-Security configurations -> *draft13-inbound-config_name* -> [Response sender] Properties -> [Additional Properties] Properties**.

Use of WS-Security Draft 13 is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the custom property.

Required	Yes
Data type	String

Value:

The value of the custom property.

Required	Yes
Data type	String

Request consumer binding [Settings]

WS-Security binding for the consumption of inbound requests from the caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *request-consumer-binding_name*.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

You use WS-Security bindings to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services. Bindings provide the information that the run-time environment needs to implement the WS-Security configuration (for example “To sign the body, use this key”).

Bindings are administered independently from any web service that uses them, so you can create a binding then apply it to many web services.

You use a *request consumer* with an inbound configuration. A *request consumer* binding consumes the requests from a client to an inbound service.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required	No
Data type	String

Name:

The name of the binding.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	String

Use defaults:

Specifies whether to use the default binding information. When this option is enabled, Web Services Security uses the default binding information instead of the custom binding information that is defined here.

Required	No
Data type	Boolean

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Token consumers
Specifies the parameters for the token consumer. The information is used only on the consumer side to process the security token. Because you can plug in a custom token consumer, you must specify a Java class name.

Key information
Specifies the related configuration that is needed to generate the key for XML digital signature or XML encryption.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Collection certificate store
Specifies a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens. The root-trusted certificates are specified in the Trust anchors panel.

Trust anchors
Specifies a list of keystore configurations that contain root-trusted certificates. These configurations are used for certificate path validation of the incoming X.509-formatted security tokens. You must create the keystore using the key tool utility. Do not use the key management utility because it does not create a keystore with the expected format.

Properties
Specifies additional properties for the configuration.

Request generator binding. [Settings]

WS-Security binding for the generation of outbound request to a target.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *request-generator-binding_name*.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

You use WS-Security bindings to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services. Bindings provide the information that the run-time environment needs to implement the WS-Security configuration (for example “To sign the body, use this key”),

Bindings are administered independently from any web service that uses them, so you can create a binding then apply it to many web services.

You use a *request generator* with an outbound configuration. A *request generator* binding generates the requests from an outbound service to a target web service.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required	No
Data type	String

Name:

The name of the binding.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

Required	Yes
Data type	String

Use defaults:

Specifies whether to use the default binding information. When this option is enabled, Web Services Security uses the default binding information instead of the custom binding information that is defined here.

Required	No
Data type	Boolean

Web Services Security namespace:

Specifies the namespace that is used by Web Services Security to send a request. However, this field configures only the name space value and does not enforce the semantics of the specification that is related to the namespace. Web Services Security uses the processing semantic only in draft 13 of the OASIS specification.

Required	Yes
Data type	drop-down list

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Token generators
Specifies the parameters for the token generator. The information is used only on the generator side to generate the security token. Because you can plug in a custom token generator, you must specify a Java class name.

Key information
Specifies the related configuration that is needed to generate the key for XML digital signature or XML encryption.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Collection certificate store
Specifies a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens. The root-trusted certificates are specified in the Trust anchors panel.

Properties
Specifies additional properties for the configuration.

Request receiver [Settings]

Draft 13 WS-Security binding for the consumption of inbound requests from the caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *request-receiver-binding_name*.

This panel is one of a set of panels that allow you to configure the service integration bus in accordance with WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required	No
Data type	String

Name:

The name of the binding.

Required	Yes
Data type	String

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for

X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Trust anchors
Specifies a list of keystore configurations that contain root-trusted certificates. These configurations are used for certificate path validation of the incoming X.509-formatted security tokens. You must create the keystore using the key tool utility. Do not use the key management utility because it does not create a keystore with the expected format.

Collection certificate store
Specifies a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens. The root-trusted certificates are specified in the Trust anchors panel.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Trusted ID evaluators
Specifies a list of trusted identity (ID) evaluators that determine whether the identity-asserting authority is trusted. You can use trusted ID evaluators for backward compatibility with Version 5 applications. However, it is recommended that you use a login module instead.

Login mappings
Specifies a list of configurations for validating security tokens within incoming messages.

Request sender [Settings]

Draft 13 WS-Security binding for the generation of requests to an outbound target.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *request-sender-binding_name*.

This panel is one of a set of panels that allow you to configure the service integration bus in accordance with WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required	No
Data type	String

Name:

The name of the binding.

Required	Yes
Data type	String

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Login binding
Specifies the configuration that is used for sending the security tokens within the messages.

Required confidentiality [Collection]

Specifies the confidentiality constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be encrypted, and the message parts to which attached encrypted Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the required confidentiality element.

Usage Indicates the assertion of the required confidentiality constraint.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Required confidentiality [Collection]

Specifies the confidentiality constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be encrypted, and the message parts to which attached encrypted Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the required confidentiality element.

Usage Indicates the assertion of the required confidentiality constraint.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Required confidentiality [Settings]

Specifies the confidentiality constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be encrypted, and the message parts to which attached encrypted Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality -> *required-confidentiality_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the required confidentiality element.

Required Yes
Data type String

Usage:

Indicates the assertion of the required confidentiality constraint.

Required Yes
Data type drop-down list
Range

Optional
Both messages that meet or do not meet the required integrity constraint are accepted.

Required
The required integrity constraint must be met by the incoming message.

Additional Properties

Message parts
Specifies parts of the message affected by this required confidentiality constraint.

Nonce
Specifies the encrypted Nonce elements which must be present in the consumed message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the encrypted time stamp elements which must be present in the consumed message, and what parts of the message they must be attached to.

Required confidentiality [Settings]

Specifies the confidentiality constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be encrypted, and the message parts to which attached encrypted Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality -> *confidentiality_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the required confidentiality element.

Required Yes
Data type String

Usage:

Indicates the assertion of the required confidentiality constraint.

Required Yes
Data type drop-down list
Range

Optional
Both messages that meet or do not meet the required integrity constraint are accepted.

Required
The required integrity constraint must be met by the incoming message.

Additional Properties

Message parts
Specifies parts of the message affected by this required confidentiality constraint.

Nonce
Specifies the encrypted Nonce elements which must be present in the consumed message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the encrypted time stamp elements which must be present in the consumed message, and what parts of the message they must be attached to.

Required integrity [Collection]

Specifies the integrity constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be digitally signed, and the message parts to which attached digitally signed Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the required integrity element.

Usage Indicates the assertion of the required integrity constraint.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Required integrity [Collection]

Specifies the integrity constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be digitally signed, and the message parts to which attached digitally signed Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the required integrity element.

Usage Indicates the assertion of the required integrity constraint.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Required integrity [Settings]

Specifies the integrity constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be digitally signed, and the message parts to which attached digitally signed Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity -> *required-integrity_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the required integrity element.

Required Yes
Data type String

Usage:

Indicates the assertion of the required integrity constraint.

Required Yes
Data type drop-down list
Range

Optional
Both messages that meet or do not meet the required integrity constraint are accepted.

Required
The required integrity constraint must be met by the incoming message.

Additional Properties

Message parts
Specifies parts of the message affected by this required integrity constraint.

Nonce
Specifies the digitally signed Nonce elements which must be present in the consumed message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the digitally signed time stamp elements which must be present in the consumed message, and what parts of the message they must be attached to.

Required integrity [Settings]

Specifies the integrity constraints consumed messages must meet. This includes specifying which message parts within the incoming message must be digitally signed, and the message parts to which attached digitally signed Nonce and time stamp elements are expected.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity -> *integrity_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the required integrity element.

Required Yes
Data type String

Usage:

Indicates the assertion of the required integrity constraint.

Required Yes

Data type
drop-down list

Range

Optional
Both messages that meet or do not meet the required integrity constraint are accepted.

Required
The required integrity constraint must be met by the incoming message.

Additional Properties

Message parts
Specifies parts of the message affected by this required integrity constraint.

Nonce
Specifies the digitally signed Nonce elements which must be present in the consumed message, and what parts of the message they must be attached to. Nonce is a randomly generated value.

Time stamp
Specifies the digitally signed time stamp elements which must be present in the consumed message, and what parts of the message they must be attached to.

Required security token [Collection]

Specifies accepted stand-alone security tokens within a consumed message. Stand-alone security tokens are those not already used for signature or encryption. Defining a required security token means that messages containing a token of that type will be processed according to the usage assertion. The security token will not be used for authentication unless it is also specified within a caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Security Token.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the security token.

Usage Indicates the assertion of the required security token constraint.

URI Specifies the namespace URI of the security token.

Local name
Specifies the local name of the security token.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Required security token [Collection]

Specifies accepted stand-alone security tokens within a consumed message. Stand-alone security tokens are those not already used for signature or encryption. Defining a required security token means that messages containing a token of that type will be processed according to the usage assertion. The security token will not be used for authentication unless it is also specified within a caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Security Token.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the security token.

Usage Indicates the assertion of the required security token constraint.

URI Specifies the namespace URI of the security token.

Local name
Specifies the local name of the security token.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Required security token [Settings]

Specifies accepted stand-alone security tokens within a consumed message. Stand-alone security tokens are those not already used for signature or encryption. Defining a required security token means that messages containing a token of that type will be processed according to the usage assertion. The security token will not be used for authentication unless it is also specified within a caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Security Token -> *required-security-token_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the security token.

Required	Yes
Data type	String

URI:

Specifies the namespace URI of the security token.

This is the namespace Uniform Resource Identifier (URI) of the security token within the consumed message.

If you specify a Username token or X.509 certificate security token, you do not have to specify a URI. If you specify a custom token, enter the URI of the `QName` for the value type. If you specify Lightweight Third Party Authentication (LTPA), enter the following WebSphere Application Server predefined value type URI: `http://www.ibm.com/websphere/appserver/tokentype/5.0.2`. If you specify Lightweight Third Party Authentication propagation (LTPA_PROPAGATION), enter the following WebSphere Application Server predefined value type URI: `http://www.ibm.com/websphere/appserver/tokentype`.

Required	No
Data type	String

Local name:

Specifies the local name of the security token.

WebSphere Application Server has the following predefined **local name** value types:

Username token
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken`

X509 certificate token
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3`

 # X509 certificates in a PKIPath
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1`

A list of X509 certificates and CRLs in a PKCS#7
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7`

LTPA For Lightweight Third Party Authentication, the **local name** value type is LTPA.

LTPA_PROPAGATION
For Lightweight Third Party Authentication token propagation, the **local name** value type is LTPA_PROPAGATION.

Attention:

- If you enter LTPA in the **Local name** field, you must also specify the URI value http://www.ibm.com/websphere/appserver/tokentype/5.0.2 in the **URI** field.
- If you enter LTPA_PROPAGATION in the **Local name** field, you must also specify the URI value http://www.ibm.com/websphere/appserver/tokentype in the **URI** field.
- If you enter any of the other predefined local name value types, you can leave the **URI** field blank. For example, to specify “Username token”, enter http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken in the **Local name** field and do not enter a value in the **URI** field.

- If you specify a custom value type for a custom token, you must specify the local name and the URI of the Quality name (QName) of the value type. For example, you might enter Custom in the **Local name** field, and http://www.ibm.com/custom in the **URI** field.

Required Yes
Data type String

Usage:

Indicates the assertion of the required security token constraint.

Required Yes
Data type drop-down list
Range

Optional
Both messages that meet or do not meet the required integrity constraint are accepted.

Required
The required integrity constraint must be met by the incoming message.

Required security token [Settings]

Specifies accepted stand-alone security tokens within a consumed message. Stand-alone security tokens are those not already used for signature or encryption. Defining a required security token means that messages containing a token of that type will be processed according to the usage assertion. The security token will not be used for authentication unless it is also specified within a caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Security Token -> *security-token_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the security token.

Required Yes
Data type String

URI:

Specifies the namespace URI of the security token.

Required No
Data type String

Local name:

Specifies the local name of the security token.

Required Yes
Data type String

Usage:

Indicates the assertion of the required security token constraint.

Required Yes
Data type drop-down list
Range

Optional
Both messages that meet or do not meet the required integrity constraint are accepted.

Required
The required integrity constraint must be met by the incoming message.

Response consumer binding [Settings]

WS-Security binding for consumption of responses from outbound target.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *response-consumer-binding_name*.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

You use WS-Security bindings to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services. Bindings provide the information that the run-time environment needs to implement the WS-Security configuration (for example “To sign the body, use this key”),

Bindings are administered independently from any web service that uses them, so you can create a binding then apply it to many web services.

You use a *response consumer* with an outbound configuration. A *response consumer* binding consumes the responses from a target web service to an outbound service.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required	No
Data type	String

Name:

The name of the binding.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required	Yes
Data type	String

Use defaults:

Specifies whether to use the default binding information. When this option is enabled, Web Services Security uses the default binding information instead of the custom binding information that is defined here.

Required	No
Data type	Boolean

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Token consumers
Specifies the parameters for the token consumer. The information is used only on the consumer side to process the security token. Because you can plug in a custom token consumer, you must specify a Java class name.

Key information
Specifies the related configuration that is needed to generate the key for XML digital signature or XML encryption.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Collection certificate store
Specifies a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens. The root-trusted certificates are specified in the Trust anchors panel.

Trust anchors
Specifies a list of keystore configurations that contain root-trusted certificates. These configurations are used for certificate path validation of the incoming X.509-formatted security tokens. You must create the keystore using the key tool utility. Do not use the key management utility because it does not create a keystore with the expected format.

Properties
Specifies additional properties for the configuration.

Response generator binding configuration [Settings]

WS-Security binding for generation of responses to caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *response-generator-binding_name*.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

You use WS-Security bindings to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services. Bindings provide the information that the run-time environment needs to implement the WS-Security configuration (for example “To sign the body, use this key”),

Bindings are administered independently from any web service that uses them, so you can create a binding then apply it to many web services.

You use a *response generator* with an inbound configuration. A *response generator* binding generates the responses from an inbound service to a client.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required No

Data type String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required No
Data type String

Name:

The name of the binding.

This name must be unique, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

Required Yes
Data type String

Use defaults:

Specifies whether to use the default binding information. When this option is enabled, Web Services Security uses the default binding information instead of the custom binding information that is defined here.

Required No
Data type Boolean

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Token generators
Specifies the parameters for the token generator. The information is used only on the generator side to generate the security token. Because you can plug in a custom token generator, you must specify a Java class name.

Key information
Specifies the related configuration that is needed to generate the key for XML digital signature or XML encryption.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Collection certificate store
Specifies a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens. The root-trusted certificates are specified in the Trust anchors panel.

Properties
Specifies additional properties for the configuration.

Response receiver [Settings]

Draft 13 WS-Security binding for consumption of responses from a target.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *response-receiver-binding_name*.

This panel is one of a set of panels that allow you to configure the service integration bus in accordance with WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required	No
Data type	String

Name:

The name of the binding.

Required	Yes
Data type	String

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Trust anchors
Specifies a list of keystore configurations that contain root-trusted certificates. These configurations are used for certificate path validation of the incoming X.509-formatted security tokens. You must create the keystore using the key tool utility. Do not use the key management utility because it does not create a keystore with the expected format.

Collection certificate store
Specifies a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens. The root-trusted certificates are specified in the Trust anchors panel.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Response sender [Settings]

Draft 13 WS-Security binding for generation of responses to a caller.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings -> *response-sender-binding_name*.

This panel is one of a set of panels that allow you to configure the service integration bus in accordance with WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

WS-Security version:

Identifies the version of the WS-Security specification this configuration uses.

Required	No
Data type	String

Binding Type:

The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Required	No
Data type	String

Name:

The name of the binding.

Required	Yes
Data type	String

Additional Properties

Signing information
Specifies the configuration for the signing parameters. You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, only fill in the Certificate path field.

Encryption information
Specifies the configuration for the XML encryption and decryption parameters. If the data and key encryption algorithms are specified, the application server only accepts elements that are encrypted with those algorithms.

Key locators
Specifies a list of key locator configurations that retrieve the key for signature and encryption. You can customize a key locator class to retrieve keys from other types of repositories. The default implementation retrieves keys from a keystore.

Security Token [Collection]

Specifies stand-alone security tokens to insert into the generated message. Stand-alone security tokens are those not already used for signature or encryption. Standard and custom security tokens may be defined by URI and local name.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Security Token.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the security token

URI Specifies the namespace URI of the security token to insert.

Local Name
Specifies the local name of the security token to insert.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Security Token [Collection]

Specifies stand-alone security tokens to insert into the generated message. Stand-alone security tokens are those not already used for signature or encryption. Standard and custom security tokens may be defined by URI and local name.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Security Token.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the security token

URI Specifies the namespace URI of the security token to insert.

Local Name
Specifies the local name of the security token to insert.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Security Token [Settings]

Specifies stand-alone security tokens to insert into the generated message. Stand-alone security tokens are those not already used for signature or encryption. Standard and custom security tokens may be defined by URI and local name.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Request generator] Security Token -> *security-token_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the security token

Required	Yes
Data type	String

URI:

Specifies the namespace URI of the security token to insert.

This is the namespace Uniform Resource Identifier (URI) of the security token to be inserted into the generated message.

If you specify a Username token or X.509 certificate security token, you do not have to specify a URI. If you specify a custom token, enter the URI of the `QName` for the value type. If you specify Lightweight Third Party Authentication (LTPA), enter the following WebSphere Application Server predefined value type URI: `http://www.ibm.com/websphere/appserver/tokentype/5.0.2`. If you specify Lightweight Third Party Authentication propagation (LTPA_PROPAGATION), enter the following WebSphere Application Server predefined value type URI: `http://www.ibm.com/websphere/appserver/tokentype`.

Required	No
Data type	String

Local Name:

Specifies the local name of the security token to insert.

WebSphere Application Server has the following predefined **local name** value types:

Username token
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken`

X509 certificate token
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3`

 # X509 certificates in a PKIPath
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1`

A list of X509 certificates and CRLs in a PKCS#7
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7`

LTPA For Lightweight Third Party Authentication, the **local name** value type is LTPA.

LTPA_PROPAGATION
For Lightweight Third Party Authentication token propagation, the **local name** value type is LTPA_PROPAGATION.

Attention:

- If you enter LTPA in the **Local name** field, you must also specify the URI value http://www.ibm.com/websphere/appserver/tokentype/5.0.2 in the **URI** field.
- If you enter LTPA_PROPAGATION in the **Local name** field, you must also specify the URI value http://www.ibm.com/websphere/appserver/tokentype in the **URI** field.

- If you enter any of the other predefined local name value types, you can leave the **URI** field blank. For example, to specify “Username token”, enter http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken in the **Local name** field and do not enter a value in the **URI** field.
- If you specify a custom value type for a custom token, you must specify the local name and the URI of the Quality name (QName) of the value type. For example, you might enter Custom in the **Local name** field, and http://www.ibm.com/custom in the **URI** field.

Required	Yes
Data type	String

Security Token [Settings]

Specifies stand-alone security tokens to insert into the generated message. Stand-alone security tokens are those not already used for signature or encryption. Standard and custom security tokens may be defined by URI and local name.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Response generator] Security Token -> *security-token_name*.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the security token

Required	Yes
Data type	String

URI:

Specifies the namespace URI of the security token to insert.

Required	No
Data type	String

Local Name:

Specifies the local name of the security token to insert.

Required	Yes
Data type	String

Time stamp [Collection]

Attaches a time stamp element to the message part specified by the dialect and keyword attributes.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity -> *required-integrity_name* -> [Additional Properties] Timestamp**
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity -> *integrity_name* -> [Additional Properties] Timestamp**
- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality -> *required-confidentiality_name* -> [Additional Properties] Timestamp**
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality -> *confidentiality_name* -> [Additional Properties] Timestamp**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Name The name of the time stamp element.

Dialect
The expression dialect to use.

Keyword
The message part to attach the time stamp element to, specified in a way defined by the chosen dialect.

Expires
The expiration time of the time stamp, defined as an xsd:Duration type.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Time stamp [Settings]

Attaches a time stamp element to the message part specified by the dialect and keyword attributes.

To view this page in the console, click one of the following paths:

- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required integrity -> *required-integrity_name* -> [Additional Properties] Timestamp -> *timestamp_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required integrity -> *integrity_name* -> [Additional Properties] Timestamp -> *timestamp_name***

- **Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Required Confidentiality -> *required-confidentiality_name* -> [Additional Properties] Timestamp -> *timestamp_name***
- **Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Required Confidentiality -> *confidentiality_name* -> [Additional Properties] Timestamp -> *timestamp_name***

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the time stamp element.

Required	Yes
Data type	String

Dialect:

The expression dialect to use.

Required	Yes
Data type	drop-down list

Keyword:

The message part to attach the time stamp element to, specified in a way defined by the chosen dialect.

When the `http://www.ibm.com/websphere/webservices/wssecurity/dialect-was` dialect value is selected, the following are valid keyword values:

action Specifies the wsa:Action element.

body Specifies the SOAP body element.

dsigkey
Specifies the key information element, which is used for digital signature.

enckey
Specifies the ds:KeyInfo element, which is used for encryption.

messageid
Specifies the wsa:MessageID element.

relatesto
Specifies the wsa:RelatesTo element.

securitytoken
Specifies any security token elements, for example the wsse:BinarySecurityToken element.

timestamp
Specifies the wsu:Timestamp element. This element determines whether the message is valid based upon the time that the message is sent and then received.

to Specifies the wsa:To element.

When the `http://www.w3.org/TR/1999/REC-xpath-1999116` dialect value is selected, then the keyword value can be any valid XPath expression that points to a part of the message. For example:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Envelope']
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/' and local-name()='Body']
```

Required	Yes
Data type	String

Expires:

The expiration time of the time stamp, defined as an xsd:Duration type.

The expires value is defined as a type of xsd:Duration, and the format must match the following regular expression:

```
-?P([0-9]+Y)?([0-9]+M)?([0-9]+D)?(T([0-9]+H)?([0-9]+M)?([0-9]+(\\.[0-9]*)?S)?)
```

For example, to specify a timestamp expiration of three minutes, enter PT3M.

Required	Yes
Data type	String

Trust Method [Settings]

Defines a trust method used to validate the identity of a trusted intermediary asserting an ID on a downstream message. When a trust method is configured, the security token defined by the caller is expected to contain an identity to be asserted.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-inbound-config_name* -> [Request consumer] Caller -> *caller_name* -> [Additional Properties] Trust method.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Trust any:

If trust any is selected then all upstream intermediaries will be trusted by this consumer. This should only be selected if you are certain that all upstream intermediaries are trusted. Selecting trust any will automatically override all other attributes of this trust method.

If you do not select the **Trust any** check box, but you specify a value for any other field on this panel, then WS-Security identity assertion is enabled.

Required	No
Data type	Boolean

Name:

The name of the trust method.

There are two valid predefined names:

- `BasicAuth` (for basic authentication).
- `Signature`.

Required No
Data type String

Part:

Specifies the name of the required integrity or required confidentiality part within the message to be used to validate the intermediary.

Required No
Data type drop-down list

URI:

Specifies the URI of the security token to use to validate the intermediary.

If you specify `BasicAuth` or `Signature` as the trust method, you do not have to specify this option. If you specify a custom token, enter the URI of the `QName` for the value type.

Required No
Data type String

Local Name:

Specifies the local name of the security token to use to validate the intermediary.

If you enter a value in the **Local Name** field, you must define a trusted ID evaluator for the token consumer that is associated with this token.

WebSphere Application Server has the following predefined **local name** value types:

BasicAuth
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken`

Signature
`http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3`

Attention:

- If you enter one of the predefined **local name** value types, you can leave the **URI** field blank. For example, to specify “BasicAuth”, enter `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken` in the **Local name** field and do not enter a value in the **URI** field.
- If you specify a custom value type for a custom token, you must specify the local name and the URI of the Quality name (QName) of the value type. For example, you might enter `Custom` in the **Local name** field, and `http://www.ibm.com/custom` in the **URI** field.

Required No
Data type String

Additional Properties

Properties
Properties associated with the trust method.

Trust Method [Settings]

Defines a trust method used to validate the identity of a trusted intermediary asserting an ID on a downstream message. When a trust method is configured, the security token defined by the caller is expected to contain an identity to be asserted.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations -> *v1-outbound-config_name* -> [Response consumer] Caller > ***caller_name*** **-> [Additional Properties] Trust method**.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Trust any:

If trust any is selected then all upstream intermediaries will be trusted by this consumer. This should only be selected if you are certain that all upstream intermediaries are trusted. Selecting trust any will automatically override all other attributes of this trust method.

Required	No
Data type	Boolean

Name:

The name of the trust method.

Required	Yes
Data type	String

Part:

Specifies the name of the required integrity or required confidentiality part within the message to be used to validate the intermediary.

Required	Yes
Data type	drop-down list

URI:

Specifies the URI of the security token to use to validate the intermediary.

Required	Yes
Data type	String

Local Name:

Specifies the local name of the security token to use to validate the intermediary.

Required	Yes
Data type	String

Additional Properties

Properties

Properties associated with the trust method.

WS-Security bindings [Collection]

WS-Security bindings for consumption and generation of requests and responses.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security bindings.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

Alternatively, you can configure the bus in accordance with the previous WS-Security specification, WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

You use WS-Security bindings to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target web services. Bindings provide the information that the run-time environment needs to implement the WS-Security configuration (for example “To sign the body, use this key”),

Bindings are administered independently from any web service that uses them, so you can create a binding then apply it to many web services. However, the security requirements for an inbound service (which acts as a target web service) are significantly different to those required for an outbound service (which acts as a client). Consequently, bindings are further divided into sub-types:

For WS-Security Version 1.0:

- *request consumer*, for use when consuming requests from a client to an inbound service.
- *request generator*, for use when generating requests from an outbound service to a target web service.
- *response consumer*, for use when consuming responses from a target web service to an outbound service.
- *response generator*, for use when generating responses from an inbound service to a client.

For WS-Security Draft 13:

- *request receiver*, for use when receiving requests from a client to an inbound service.
- *request sender*, for use when sending requests from an outbound service to a target web service.
- *response receiver*, for use when receiving responses from a target web service to an outbound service.
- *response sender*, for use when sending responses from an inbound service to a client.

Name The name of the WS-Security binding.

Binding Type
The type of binding. This is one of request consumer, request generator, response consumer and response generator.

Security version
Identifies the version of the WS-Security specification this configuration uses.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

WS-Security configurations [Collection]

WS-Security configurations for inbound and outbound services.

To view this page in the console, click the following path:

Service integration -> Web services -> WS-Security configurations.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

You can configure the service integration bus for secure transmission of SOAP messages by using tokens, keys, signatures and encryption in accordance with the Web Services Security (WS-Security) 1.0 specification.

Alternatively, you can configure the bus in accordance with the previous WS-Security specification, WS-Security Draft 13 (also known as the Web Services Security Core Specification). However, use of the WS-Security Draft 13 specification is deprecated, and you should only use it to allow continued use of an existing web services client application that has been written to the WS-Security Draft 13 specification.

You use WS-Security configurations to secure the SOAP messages that pass between service requesters (clients) and inbound services, and between outbound services and target Web services. Configurations specify the level of security that you require (for example “The body must be signed”). This level of security is then implemented through the run-time information contained in a WS-Security binding.

Configurations are administered independently from any web service that uses them, so you can create a configuration then apply it to many web services. However, the security requirements for an inbound service (which acts as a target web service) are significantly different to those required for an outbound service (which acts as a client). Consequently, configurations are further divided by service type (inbound or outbound).

Name The name of the WS-Security configuration.

Service type
The type of service the WS-Security configuration applies to.

Security version
Identifies the version of the WS-Security specification this configuration uses.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

SIBWebServices command group for the AdminTask object

Use command scripts to configure service integration bus-enabled web services.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Before and immediately after performing administrative commands that carry out publication to UDDI, save the configuration by using either the `AdminConfig.save()` command or an equivalent command from within the administrative console. This ensures consistency between what is published to UDDI and what is recorded in the service integration bus configuration as having been published to UDDI.

Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('SIBWebServices')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

The following administrative commands are available in this command group.

- “createSIBWSOutboundService command” on page 2934
- “deleteSIBWSOutboundService command” on page 2935
- “addSIBWSOutboundPort command” on page 2936
- “removeSIBWSOutboundPort command” on page 2938
- “setDefaultSIBWSOutboundPort command” on page 2939
- “createSIBWSInboundService command” on page 2940
- “deleteSIBWSInboundService command” on page 2941
- “addSIBWSInboundPort command” on page 2943
- “removeSIBWSInboundPort command” on page 2944
- “refreshSIBWSOutboundServiceWSDL command” on page 2945
- “refreshSIBWSInboundServiceWSDL command” on page 2946

- “publishSIBWSInboundService command” on page 2947
- “unpublishSIBWSInboundService command” on page 2948
- “createSIBWSEndpointListener command” on page 2949
- “deleteSIBWSEndpointListener command” on page 2951
- “connectSIBWSEndpointListener command” on page 2952
- “disconnectSIBWSEndpointListener command” on page 2953

createSIBWSOutboundService command

Use the createSIBWSOutboundService command to create a new service integration bus-enabled web services outbound service configuration.

You can create a new outbound service configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Making an externally-hosted web service available internally” on page 2783.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a new OutboundService object, that represents a protocol attachment to a service provider. When you run this command you must identify a single service element within a WSDL document.

Target object

ObjectName of the messaging bus within which the service is created.

If the WSDL is to be retrieved through a proxy, the server on which the command is running must have the system properties that identify the proxy server set correctly. If the proxy requires authentication, then the user ID and password can be set as parameters on the command.

After you have run this command, you use other commands to further configure the service. For example, to add an outbound port.

Required parameters

-name
The outbound service name.

-wsdlLocation
The location of the service provider WSDL file.

This is either a web address or the service-specific part of a UDDI service key. If you specify a UDDI reference, the WSDL location is assumed to be a UDDI service key.

Here is an example of a full UDDI service key:

```
uddi:blade108node01cell:blade108node01:server1:default:6e3d106e-5394-44e3-be17-aca728ac1791
```

The service-specific part of this key is the final part:

```
6e3d106e-5394-44e3-be17-aca728ac1791
```

Conditional parameters

-wsdlServiceName
The name of the service within the WSDL. Only required if the service provider WSDL contains more than one service, or the WSDL is located through a UDDI registry.

-wsdlServiceNamespace
The namespace of the service within the WSDL. Only required if the service provider WSDL contains more than one service, or the WSDL is located through a UDDI registry, or the service is not in the default namespace for the WSDL document.

Optional parameters

-uddiReference
If you specified a UDDI service key as the WSDL location, supply the UDDI reference for the target UDDI registry.

-destination
The name of the service destination.

Note: The command creates the service destination. If a destination with the specified or default name already exists, the command fails.

-userId
The user ID that you use to retrieve the WSDL.

-password
The password that you use to retrieve the WSDL.

Example

- Using Jython:

```
outService = AdminTask.createSIBWSOutboundService(bus, ["-name", "MyService",
 "-wsdlLocation", "http://myserver.com/MyService.wsdl"])
```

- Using Jacl:

```
set outService [$AdminTask createSIBWSOutboundService $bus {-name "MyService"
 -wsdlLocation "http://myserver.com/MyService.wsdl"}]
```

deleteSIBWSOutboundService command

Use the deleteSIBWSOutboundService command to delete a service integration bus-enabled web services outbound service configuration.

You can delete an outbound service configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting outbound service configurations” on page 2788.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes an OutboundService object, along with any associated OutboundPort objects.

Target object

ObjectName of the OutboundService object to be deleted.

The service and port destinations are deleted. Any messages on the service and port destinations are either deleted or sent to an exception destination as specified in the policy for the messaging bus.

Resources associated with the OutboundService and OutboundPorts (JAX-RPC handler lists, WS-Security configuration) are dissociated from the OutboundService and OutboundPorts, but are not themselves deleted.

Parameters

None.

Example

- Using Jython:

  ```
  AdminTask.deleteSIBWSOutboundService(outService)
  ```

- Using Jacl:

  ```
  $AdminTask deleteSIBWSOutboundService $outService
  ```

addSIBWSOutboundPort command

Use the addSIBWSOutboundPort command to add a service integration bus-enabled web services outbound port.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds the configuration for an OutboundPort to an OutboundService.

Target object

ObjectName of the OutboundService with which the port is to be associated.

If an OutboundPort for the OutboundService by the given name already exists, or the name of the port does not correspond to a port in the service element of the service provider WSDL (as defined on the OutboundService) the command fails.

The command creates the port destination. If a destination with the specified or default name already exists, the command fails.

If the WSDL is to be retrieved through a proxy server, the server on which the command is running must have the system properties that identify the proxy server set correctly. In addition, if the proxy server requires authentication, then the user ID and password can be set as parameters on the command.

If this is the first port for the OutboundService, then it is set as the default, and the service destination default routing is set to point to the port destination.

Required parameters

-name
The name of the port in the service provider WSDL.

Conditional parameters

-node
The node in which the port destination is localized.

-server
The server in which the port destination is localized.

Optional parameters

-destination
The name of the port destination.

-userId
The user ID that you use to retrieve the WSDL.

Note: This parameter is no longer required. If you have existing scripts that provide this parameter, they will continue to work.

-password
The password that you use to retrieve the WSDL.

Note: This parameter is no longer required. If you have existing scripts that provide this parameter, they will continue to work.

Example

- Using Jython:

```
outPort = AdminTask.addSIBWSOutboundPort(outService, ["-name", "MyServiceSoap",
 "-node", "MyNode", "-server", "server1"])
```

- Using Jacl:

```
set outPort [$AdminTask addSIBWSOutboundPort $outService {-name "MyServiceSoap"
 -node "MyNode" -server "server1"}]
```

removeSIBWSOutboundPort command

Use the removeSIBWSOutboundPort command to remove a service integration bus-enabled web services outbound port.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes the configuration for an OutboundPort.

Target object

ObjectName of the OutboundPort object that is to be removed.

The port destination is deleted. Any messages on the port destination are either deleted or sent to an exception destination as specified in the policy for the messaging bus.

If the port that is removed is the default for the OutboundService, then one of the remaining ports (if any) is chosen to be the default, and the default routing on the service destination is updated. If there are no more ports, the default is cleared from the OutboundService and the service destination.

Resources associated with the OutboundPort (JAX-RPC handler lists, WS-Security configuration) are dissociated from the OutboundPort, but not deleted.

Parameters

None.

Example

- Using Jython:

```
AdminTask.removeSIBWSOutboundPort(outPort)
```

- Using Jacl:

```
$AdminTask removeSIBWSOutboundPort $outPort
```

setDefaultSIBWSOutboundPort command

Use the setDefaultSIBWSOutboundPort command to set the default service integration bus-enabled web services outbound port.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBWebServices')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command updates the default outbound port for an outbound service.

Target object

ObjectName of the OutboundService object whose default port is to be updated.

The default port is updated for the OutboundService, and the default routing on the service destination is updated to point at the port destination.

Required parameters

-name
The name of the port to be set as the default.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
AdminTask.setDefaultSIBWSOutboundPort(outService, ["-name", "MyServiceSoap"])
```

- Using Jacl:

```
$AdminTask setDefaultSIBWSOutboundPort $outService {-name "MyServiceSoap"}
```

createSIBWSInboundService command

Use the createSIBWSInboundService command to create a new service integration bus-enabled web services inbound service configuration.

You can create a new inbound service configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Making an internally-hosted service available as a web service” on page 2778.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBWebServices')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a new InboundService object that represents a protocol attachment to be used by service requesters. When you run this command you identify a single service element within a template WSDL document, and an existing service destination.

Target object

ObjectName of the service integration bus within which the service is created.

If the WSDL is to be retrieved through a proxy server, the server on which the command is running must have the system properties that identify the proxy server set correctly. If the proxy server requires authentication, then the user ID and password can be set as parameters on the command.

After you have run this command, you can use other commands to further configure the service. For example, you can add an inbound port.

Required parameters

-name
The inbound service name. This cannot be longer than 250 characters.

-destination
The name of the service destination. If the specified destination does not exist, the command fails.

-wsdlLocation
The location of the template WSDL file.

This is either a web address or the service-specific part of a UDDI service key. If you specify a UDDI reference, the WSDL location is assumed to be a UDDI service key.

Here is an example of a full UDDI service key:

```
uddi:blade108node01cell:blade108node01:server1:default:6e3d106e-5394-44e3-be17-aca728ac1791
```

The service-specific part of this key is the final part:

```
6e3d106e-5394-44e3-be17-aca728ac1791
```

Conditional parameters

-wsdlServiceName
The name of the service within the template WSDL. Only required if the template WSDL contains more than one service, or the WSDL is located through a UDDI registry

-wsdlServiceNamespace
The namespace of the service within the WSDL. Only required if the template WSDL contains more than one service, or the WSDL is located through a UDDI registry, or the service is not in the default namespace for the WSDL document.

Optional parameters

-uddiReference
If you specified a UDDI service key as the template WSDL location, supply the UDDI reference for the target UDDI registry.

-userId
The user ID that you use to retrieve the WSDL.

-password
The password that you use to retrieve the WSDL.

Example

- Using Jython:

```
inService = AdminTask.createSIBWSInboundService(bus, ["-name", "MyService",
 "-destination", "destName",
 "-wsdlLocation", "http://myserver.com/MyService.wsdl"] )
```

- Using Jacl:

```
set inService [$AdminTask createSIBWSInboundService $bus {-name "MyService"
 -destination $destName
 -wsdlLocation "http://myserver.com/MyService.wsdl"}]
```

deleteSIBWSInboundService command

Use the deleteSIBWSInboundService command to delete a service integration bus-enabled web services inbound service configuration.

You can delete an inbound service configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting inbound services configurations” on page 2783.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes an InboundService object, along with any associated InboundPort objects.

Target object

ObjectName of the InboundService object to be deleted.

Resources associated with the InboundService and its InboundPorts (JAX-RPC handler lists, WS-Security configuration) are dissociated from the InboundService and InboundPorts but are not themselves deleted.

The optional parameters user ID and password allow the unpublishing of WSDL from UDDI registries through an authenticating proxy server. This command fails if different UDDIPublication objects defined for the InboundService need different user IDs or passwords to get the appropriate access.

Required parameters

None.

Conditional parameters

None.

Optional parameters

-userId
The user ID that you use to interact with UDDI registries.

-password
The password that you use to interact with UDDI registries.

Example

- Using Jython:

  ```
  AdminTask.deleteSIBWSInboundService(inService)
  ```

- Using Jacl:

  ```
  $AdminTask deleteSIBWSInboundService $inService
  ```

addSIBWSInboundPort command

Use the addSIBWSInboundPort command to add a service integration bus-enabled web services inbound port.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds the configuration for an InboundPort to an InboundService.

If the port name is already in use by another InboundPort for the InboundService, or the specified endpoint listener does not exist, the command fails.

If the templatePort is specified but does not exist in the InboundService template WSDL, the command fails.

If there is no BusConnectionProperty for the InboundService bus, then one is created with the default reply destination name.

Target object

ObjectName of the InboundService to which the port is to be added.

Required parameters

-name
The name of the port.

-endpointListener
The name of the associated endpoint listener.

Conditional parameters

-node
The node in which the endpoint listener is located.

-server
The server in which the endpoint listener is located.

Optional parameters

-templatePort
The name of the port in the template WSDL to use as a basis for this port binding.

Example

- Using Jython:

```
inPort = AdminTask.addSIBWSInboundPort(inService, ["-name", "MyServiceSoap",
 "-endpointListener", "SOAPHTTP1", "-node", "MyNode", "-server", "server1"] )
```

- Using Jacl:

```
set inPort [$AdminTask addSIBWSInboundPort $inService {-name "MyServiceSoap"
 -endpointListener "SOAPHTTP1" -node "MyNode" -server "server1"}]
```

removeSIBWSInboundPort command

Use the removeSIBWSInboundPort command to remove a service integration bus-enabled web services inbound port.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes the configuration for an InboundPort.

Target object

ObjectName of the InboundPort object to be removed.

Resources associated with the InboundPort (JAX-RPC handler lists, WS-Security configuration) are dissociated from the InboundPort, but are not themselves deleted.

Parameters

None.

Example

- Using Jython:

```
AdminTask.removeSIBWSInboundPort(inPort)
```

- Using Jacl:

```
$AdminTask removeSIBWSInboundPort $inPort
```

refreshSIBWSOutboundServiceWSDL command

Use the refreshSIBWSOutboundServiceWSDL command to refresh a service integration bus-enabled web services outbound service WSDL file.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBWebServices')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command loads the WSDL document from the OutboundService WSDLLocation parameters, and locates the WSDLLocation-specified service element.

Target object

ObjectName of the OutboundService object.

If the service element is not present, the command fails.

The OutboundPorts must be a subset of the ports in the loaded WSDL document, otherwise the command fails.

If the WSDL is to be retrieved through a proxy, the server on which the command is running must have the system properties that identify the proxy server set correctly. If the proxy requires authentication, then the user ID and password can be set as parameters on the command.

Required parameters

None.

Conditional parameters

None.

Optional parameters

-userId
The user ID that you use to retrieve the WSDL.

-password
The password that you use to retrieve the WSDL.

Example

- Using Jython:
  ```
  AdminTask.refreshSIBWSOutboundServiceWSDL(outService)
  ```
- Using Jacl:
  ```
  $AdminTask refreshSIBWSOutboundServiceWSDL $outService
  ```

refreshSIBWSInboundServiceWSDL command

Use the refreshSIBWSInboundServiceWSDL command to refresh a service integration bus-enabled web services inbound service WSDL file.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBWebServices')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command loads the WSDL document from the InboundService WSDLLocation parameters, and locates the WSDLLocation-specified service element.

Target object

ObjectName of the InboundService object.

If the service element is not present, the command fails.

All templatePortName values in InboundPorts for the InboundService must correspond to ports in the loaded WSDL document, otherwise the command fails.

If the WSDL is to be retrieved through a proxy, the server on which the command is running must have the system properties that identify the proxy server set correctly. If the proxy requires authentication, then the user ID and password can be set as parameters on the command.

Required parameters

None.

Conditional parameters

None.

Optional parameters

-userId
The user ID that you use to retrieve the WSDL.

-password
The password that you use to retrieve the WSDL.

Example

- Using Jython:
  ```
  AdminTask.refreshSIBWSInboundServiceWSDL(inService)
  ```
- Using Jacl:
  ```
  $AdminTask refreshSIBWSInboundServiceWSDL $inService
  ```

publishSIBWSInboundService command

Use the publishSIBWSInboundService command to publish to UDDI a service integration bus-enabled web services inbound service WSDL file.

Before and immediately after performing this command, save the configuration by using either the `AdminConfig.save()` command or an equivalent command from within the administrative console. This ensures consistency between what is published to UDDI and what is recorded in the service integration bus configuration as having been published to UDDI.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBWebServices')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command publishes the WSDL document for the InboundService, with all its ports, to the registry and business defined by the UDDIPublication object.

Target object

ObjectName of the InboundService object.

If the UDDI publish operation succeeds, the service key in the UDDIPublication object is updated and a warning message is produced indicating that the service is successfully published to UDDI, but that without a save of the configuration the system is in an inconsistent state.

If the UDDI publish operation fails, the service key is not updated and an error message is produced indicating that the publish operation failed.

If the UDDI publish operation succeeds, an information message is produced that contains sufficient details for the administrator to independently find the service in the UDDI registry if that becomes necessary.

If the WSDL is to be published through a proxy, the server on which the command is running must have the system properties that identify the proxy server set correctly. If the proxy requires authentication, then the user ID and password can be set as parameters on the command.

Required parameters

-uddiPublication
The name of the UDDI publication property for this service.

Conditional parameters

None.

Optional parameters

-userId
The user ID that you use to retrieve the WSDL.

-password
The password that you use to retrieve the WSDL.

Example

- Using Jython:

```
AdminTask.publishSIBWSInboundService(inService, ["-uddiPublication", "MyUddi"])
```

- Using Jacl:

```
$AdminTask publishSIBWSInboundService $inService {-uddiPublication "MyUddi"}
```

unpublishSIBWSInboundService command

Use the unpublishSIBWSInboundService command to remove from UDDI a service integration bus-enabled web services inbound service WSDL file.

Before and immediately after performing this command, save the configuration by using either the `AdminConfig.save()` command or an equivalent command from within the administrative console. This ensures consistency between what is published to UDDI and what is recorded in the service integration bus configuration as having been published to UDDI.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBWebServices')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

This command removes the WSDL document for the InboundService, with all its ports, from the registry and business defined by the UDDIPublication object.

Target object

ObjectName of the InboundService object.

If the UDDIPublication object has no service key, the command fails.

If the UDDI removal operation succeeds, the service key in the UDDIPublication object is cleared and a warning message is produced indicating that the service is successfully removed from UDDI, but that without a save of the configuration the system is in an inconsistent state and the service must be republished to UDDI by using the GUI or publishToUDDI command.

If the UDDI removal operation fails because the service key is not found, the service key is cleared and a warning message is produced indicating that the service is not found.

If the UDDI removal operation fails for any other reason, the service key is not cleared and an error message is produced indicating that the removal failed.

If the WSDL is to be published through a proxy, the server on which the command is running must have the system properties that identify the proxy server set correctly. If the proxy requires authentication, then the user ID and password can be set as parameters on the command.

Required parameters

-uddiPublication
The name of the UDDI publication property for this service.

Conditional parameters

None.

Optional parameters

-userId
The user ID that you use to retrieve the WSDL.

-password
The password that you use to retrieve the WSDL.

Example

- Using Jython:

```
AdminTask.unpublishSIBWSInboundService(inService, ["-uddiPublication", "MyUddi"])
```

- Using Jacl:

```
$AdminTask unpublishSIBWSInboundService $inService {-uddiPublication "MyUddi"}
```

createSIBWSEndpointListener command

Use the createSIBWSEndpointListener command to create a new service integration bus-enabled web services endpoint listener configuration.

For every server that is to host an endpoint listener, you must install and configure a Service Data Objects (SDO) repository on the server.

If you want to change the default HTTP endpoint listener security role, do so before you configure the SOAP over HTTP endpoint listener.

Before you configure a SOAP over JMS endpoint listener, you should configure the associated JMS resources.

You can configure any number of endpoint listeners with values of your own choosing, including the values given in “Example values for endpoint listener configuration” on page 2792.

You can create a new endpoint listener configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Creating a new endpoint listener configuration” on page 2789.

Note: If you want to create an endpoint listener configuration for your own endpoint listener application, rather than for one of the listeners that is supplied with WebSphere Application Server, you must use the wsadmin tool.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates an endpoint listener configuration.

Target object

ObjectName of the server on which the endpoint listener is created.

The SIBWSEndpointListener object that is created has no associated SIBWSBusConnectionProperty objects. Use the administrative console to connect one or more service integration buses to this endpoint listener, as described in “Modifying an existing endpoint listener configuration” on page 2797.

Required parameters

-name

The name of the endpoint listener within the server. If you are installing your own endpoint listener application, rather than one that is supplied with WebSphere Application Server, then this name must match the name given in the endpoint listener application that you have installed (that is, the *display name* of the endpoint module within the endpoint application EAR file).

-urlRoot
The root of the web address that should be used to build the endpoint addresses within WSDL documents to direct requesters to this endpoint listener.

-wsdlUrlRoot
The root of the web address for the WSDL files of the inbound services that are provided by this endpoint listener.

Conditional parameters

-earFile
The location of the endpoint listener application. Specify this parameter if you are configuring an endpoint listener other than those supplied with WebSphere Application Server.

Optional parameters

None.

Example

- Using Jython:

```
epl = AdminTask.createSIBWSEndpointListener(server, ["-name", "soaphttp1",
 "-urlRoot", "http://myserver.com/wsgwsoaphttp1",
 "-wsdlUrlRoot", "http://myserver.com/wsgwsoaphttp1"] )
```

- Using Jacl:

```
set epl [$AdminTask createSIBWSEndpointListener $server {-name "soaphttp1"
 -urlRoot "http://myserver.com/wsgwsoaphttp1"
 -wsdlUrlRoot "http://myserver.com/wsgwsoaphttp1"}]
```

deleteSIBWSEndpointListener command

Use the deleteSIBWSEndpointListener command to delete a service integration bus-enabled web services endpoint listener configuration.

You can delete an endpoint listener configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting endpoint listener configurations” on page 2799.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('SIBWebServices')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes an endpoint listener configuration.

Target object

ObjectName of the endpoint listener that is to be deleted.

The command deletes a SIBWSEndpointListener object, along with any associated SIBWSBusConnectionProperty objects.

The command fails if there are any InboundPort objects associated with the endpoint listener.

Parameters

None.

Example

- Using Jython:
  ```
  AdminTask.deleteSIBWSEndpointListener(epl)
  ```
- Using Jacl:
  ```
  $AdminTask deleteSIBWSEndpointListener $epl
  ```

connectSIBWSEndpointListener command

Use the connectSIBWSEndpointListener command to connect a service integration bus-enabled web services endpoint listener to a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('SIBWebServices')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command connects an endpoint listener to a service integration bus.

The command creates a SIBWSBusConnectionProperty object for the endpoint listener, and a reply destination. The command also sets the reply destination name in the bus connection properties.

Target object

ObjectName of the endpoint listener to be connected.

Required parameters

-bus
The name of the service integration bus to which the endpoint listener is to be connected.

Conditional parameters

None.

Optional parameters

-replyDestination
The name to use for the reply destination for this connection. If no destination name is specified, the command generates a name.

Example

- Using Jython:

```
busConn = AdminTask.connectSIBWSEndpointListener(epl, "[-bus myBus]")
```

- Using Jacl:

```
set busConn [$AdminTask connectSIBWSEndpointListener $epl {-bus "MyBus"}]
```

disconnectSIBWSEndpointListener command

Use the disconnectSIBWSEndpointListener command to disconnect a bus-enabled web services endpoint listener from a service integration bus.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available bus-enabled web services commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('SIBWebServices')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command disconnects an endpoint listener from a service integration bus.

Target object

ObjectName of the endpoint listener to be disconnected.

The command deletes the SIBWSBusConnectionProperty object for the endpoint listener, and also deletes the reply destination (if any) that is named in the bus connection properties.

Required parameters

-bus
The name of the service integration bus from which the endpoint listener is to be disconnected.

Conditional parameters

None.

Optional parameters

None.

Example

- Using Jython:

```
AdminTask.disconnectSIBWSEndpointListener(epl, ["-bus", "MyBus"])
```

- Using Jacl:

```
$AdminTask disconnectSIBWSEndpointListener $epl {-bus "MyBus"}
```

Chapter 31. Administering web services - Invocation framework (WSIF)

The Web Services Invocation Framework (WSIF) is a Web Services Description Language (WSDL)-oriented Java™ API. You use this API to invoke web services dynamically, regardless of the service implementation format (for example enterprise bean) or the service access mechanism (for example Java Message Service (JMS)). Using WSIF, you can move away from the usual web services programming model of working directly with the SOAP APIs, towards a model where you interact with representations of the services. You can therefore work with the same programming model regardless of how the service is implemented and accessed.

Administering WSIF

An overview of where and how the Web Services Invocation Framework (WSIF) is installed as part of WebSphere Application Server, and information about configuring a WSIF client to invoke a web service through JMS, and modifying the contents of the wsif.properties file.

About this task

WSIF is a thin abstraction layer between application code and the relevant invocation infrastructure.

WSIF is provided in a JAR file named `com.ibm.ws.runtime.jar`. The JAR file contains the WSIF classes, and the Java, EJB, SOAP over HTTP and SOAP over JMS providers. Additional providers are packaged as separate JAR files. When you install WebSphere Application Server, the `com.ibm.ws.runtime.jar` file is put on the WebSphere or Java virtual machine (JVM) class path.

WSIF requires no further configuration, apart from the following administrative tasks:

Procedure

- Enable a WSIF client to invoke a web service through JMS.

  Use the administrative console to configure the JMS resources (a queue destination and a queue connection factory) that are required to enable a service to be invoked through JMS by a WSIF client application.
- Modify the contents of the wsif.properties file.

  You might have to modify the contents of this file, for example to change the default SOAP provider

Enabling a WSIF client to invoke a web service through JMS

The ways in which the Web Services Invocation Framework (WSIF) interacts with the Java Message Service (JMS), and the steps to take to enable a service to be invoked through JMS by a WSIF client application.

Before you begin

This topic assumes that you chose and configured a JMS provider when you installed WebSphere Application Server (either the default messaging provider, or another provider such as the WebSphere MQ messaging provider). If not, do so now as described in Choosing a messaging provider.

About this task

Here are the ways in which WSIF interacts with JMS:
- WSIF only supports input JMS properties.
- WSIF needs two queues when invoking an operation: one for the request message and one for the reply.

- The replyTo queue is by default a temporary queue, which WSIF creates on behalf of the application. You can specify a permanent queue by setting the **JMSReplyTo** property to the JNDI name of a queue.
- WSIF uses the default values for properties set by the JMS implementation.

To enable a service to be invoked through JMS by a WSIF client application, complete the following steps:

Procedure

1. Use the administrative console to create and configure a queue connection factory and a queue destination for your chosen messaging provider.

   For more information, see "Configuring resources for the default messaging provider" on page 516, "Configuring JMS resources for the WebSphere MQ messaging provider" on page 734 or Chapter 14, "Managing messaging with a third-party or (deprecated) V5 default messaging provider," on page 933.

   Note: In WebSphere MQ and some other JMS implementations, messages are persistent by default. The WSIF replyTo temporary queue is of type temporary dynamic by default, which means that your JMS provider cannot write a persistent response message to this queue. If you are using the WebSphere MQ messaging provider, create a temporary model queue that is of type permanent dynamic, then pass this model as the **tempmodel** of your queue connection factory. This ensures that persistent messages are written to a temporary replyTo queue that is of type permanent dynamic.
2. Use the administrative console to add the new queue destination to the list of JMS destination names for your application server. Ensure that the **Initial State** is started.
3. Put the JNDI names of the queue destination and queue connection factory, as well as your JNDI configuration, in the Web Services Description Language (WSDL) file.
4. Optional: If your client is running on an application server that has been migrated from WebSphere Application Server Version 5, you might get basic authentication errors and therefore have to modify your security settings. For more information see Web Services Invocation Framework troubleshooting tips.

Configuring resources for the default messaging provider

Use the following tasks to configure JMS connection factories, activation specifications, and destinations for the default messaging provider.

About this task

Use these tasks to configure administrative JMS resources provided by the default messaging provider.

These administrative JMS resources are in addition to any temporary JMS destinations created by applications.

- List JMS resources.
- Configure a unified connection factory.
- Configure a queue connection factory.
- Configure a topic connection factory.
- Configure a queue.
- Configure a topic.
- Configure an activation specification.

Configuring JMS resources for the WebSphere MQ messaging provider

Use the WebSphere Application Server administrative console to configure activation specifications, connection factories and destinations for the WebSphere MQ JMS provider.

Before you begin

This task assumes that you are working in a mixed WebSphere Application Server and WebSphere MQ environment, and that you have decided to use the WebSphere MQ messaging provider to handle JMS

messaging between the two systems. If your business uses WebSphere MQ, and you want to integrate WebSphere Application Server messaging applications into a predominately WebSphere MQ network, the WebSphere MQ messaging provider is the natural choice. However, there can be benefits in using another provider. If you are not sure which provider combination is best suited to your needs, see Choosing messaging providers for a mixed environment.

You can configure JMS resources for the WebSphere MQ messaging provider through the administrative console as described in this task, or you can configure JMS resources for the WebSphere MQ messaging provider through the WebSphere MQ administrative commands.

About this task

Using the administrative console, you can set the scope of the WebSphere MQ messaging provider to restrict the range of resources that are displayed:

- If you set the scope to contain only WebSphere Application Server Version 6 or Version 7.0 or later nodes, you can configure JMS 1.1 resources and properties. This includes unified JMS connection factories for use by both point-to-point and publish/subscribe JMS 1.1 applications. With JMS 1.1, this approach is preferred to the domain-specific queue connection factory and topic connection factory.
- If you set the scope to contain only WebSphere Application Server Version 7.0 or later nodes, you can also configure JMS activation specifications.
- If you set the scope to a WebSphere Application Server Version 5 node, you can only configure domain-specific JMS resources, and the subset of properties that apply to WebSphere Application Server Version 5.

Note:

There are two ways of specifying the information needed by WebSphere MQ messaging provider messaging resources so that they can connect to a WebSphere MQ queue manager. It can either be specified manually, or by providing the WebSphere MQ messaging provider resource with a uniform resource locator (URL) that points to a client channel definition table (CCDT).

A CCDT is a binary file that contains information about how to create a client connection channel to one or more queue managers. The file contains information such as the hostname, port, and name of the target queue manager, as well as more advanced configuration information like the SSL attributes that should be used.

Creating WebSphere MQ messaging provider resources using CCDTs provides the following benefits:

- Flexibility, because client connection channel information is contained in a single place. If any of the information changes, such as the host name of the machine on which the WebSphere MQ queue manager resides, only the CCDT needs to be updated. When it is updated, all WebSphere MQ messaging provider resources that make use of the CCDT pick up the change.
- Reliability, because less information is needed for a CCDT there is a reduced chance of configuration errors. When using a CCDT to enter connection information, all that is required are the CCDT URL and an optional queue manager name. If you configure a WebSphere MQ messaging provider resource manually, much more information is required -- especially if you are configuring SSL.

For further information about generating a CCDT, see the WebSphere MQ information center.

Maintenance note: The WebSphere MQ messaging provider uses code provided by the WebSphere MQ resource adapter, which is automatically installed as part of the product.

Procedure

- “Creating an activation specification for the WebSphere MQ messaging provider” on page 736
- “Configuring an activation specification for the WebSphere MQ messaging provider” on page 738
- “Migrating a listener port to an activation specification for use with the WebSphere MQ messaging provider” on page 758
- “Creating a connection factory for the WebSphere MQ messaging provider” on page 759
- “Configuring a unified connection factory for the WebSphere MQ messaging provider” on page 761
- “Configuring a queue connection factory for the WebSphere MQ messaging provider” on page 789
- “Configuring a topic connection factory for the WebSphere MQ messaging provider” on page 813
- “Configuring a queue for the WebSphere MQ messaging provider” on page 840
- “Configuring a topic for the WebSphere MQ messaging provider” on page 852
- “Configuring custom properties for WebSphere MQ messaging provider JMS resources” on page 862

Managing messaging with a third-party or (deprecated) V5 default messaging provider

For messaging between application servers, most requirements are best met by either the default messaging provider or the WebSphere MQ messaging provider. However, you can instead use a third-party messaging provider (that is, use another company's product as the provider). You might want to do this, for example, if you have existing investments. For backwards compatibility with earlier releases, there is also support for the V5 default messaging provider.

Before you begin

If you are not sure which provider combination is best suited to your needs, see Types of messaging providers.

About this task

Enterprise applications in WebSphere Application Server can use asynchronous messaging through services based on Java Message Service (JMS) messaging providers and their related messaging systems. These messaging providers conform to the JMS Version 1.1 specification.

The choice of provider depends on what your JMS application needs to do, and on other factors relating to your business environment and planned changes to that environment.

Procedure

- Choose a third-party messaging provider.

  To administer a third-party messaging provider, you use either the resource adaptor (for a Java EE Connector Architecture (JCA) 1.5-compliant messaging provider) or the client (for a non-JCA messaging provider) that is supplied by the third party. You use the WebSphere Application Server administrative console to administer the activation specifications, connection factories and destinations that are within WebSphere Application Server, but you cannot use the administrative console to administer the JMS provider itself, or any of its resources that are outside of WebSphere Application Server.

  To use message-driven beans, third-party messaging providers must either provide an inbound JCA 1.5-compliant resource adapter, or (for non-JCA messaging providers) include Application Server Facility (ASF), an optional feature that is part of the JMS Version 1.1 specification.

  To work with a third-party provider, choose one of the following options:

  1. Manage messaging with a third-party JCA 1.5-compliant messaging provider.
  2. Manage messaging with a third-party non-JCA messaging provider.
- Choose the (deprecated) V5 default messaging provider.

  This deprecated provider is identical to the WebSphere Application Server Version 5 default provider. Only the name has changed. It provides backwards compatibility that enables WebSphere Application

Server Version 6 or later applications to connect to WebSphere Application Server Version 5 resources in a mixed cell. It also allows WebSphere Application Server Version 5 applications to connect to WebSphere Application Server Version 6 or later resources in a mixed cell. To configure and manage messaging to interoperate with WebSphere Application Server Version 5, see “Maintaining (deprecated) Version 5 default messaging resources” on page 959.

wsif.properties file - Initial contents

The Web Services Invocation Framework (WSIF) properties are stored in the `com.ibm.ws.runtime.jar` file, in a properties file named `wsif.properties`. You might have to modify the contents of this file, for example to change the default SOAP provider, so for reference here is a copy of the “as shipped” contents of the `wsif.properties` file.

The `com.ibm.ws.runtime.jar` file is located in the *app_server_root*`/plugins` directory, where *app_server_root* is the root directory for your installation of IBM WebSphere Application Server.

You must keep the `wsif.properties` file on the class path, so that WSIF can find it and the client administrator can use it to configure WSIF. However if you make any changes to the file, you do not replace the original copy in the `com.ibm.ws.runtime.jar` file. Instead, you save the modified version in the *app_server_root*`/lib/properties` directory.

The following code is the initial contents of the `wsif.properties` file. All the possible properties are listed and described.

```
# Two properties are used to override which WSIFProvider is selected when there
# exists multiple providers supporting the same namespace URI. These properties are:
#
#    wsif.provider.default.CLASSNAME=N
#    wsif.provider.uri.M.CLASSNAME=URI
#
# CLASSNAME is the WSIFProvider class name
# N is the number of following default wsif.provider.uri.M.CLASSNAME properties
# M is a number from 1 to N to uniquely identify each wsif.provider.uri.M.CLASSNAME
#   property key.
# For example the following two properties would override the default SOAP provider
# to be the Apache SOAP provider:
#
# wsif.provider.default.org.apache.wsif.providers.soap.ApacheSOAP.WSIFDynamicProvider_ApacheSOAP=1
# wsif.provider.uri.1.org.apache.wsif.providers.soap.ApacheSOAP.WSIFDynamicProvider_ApacheSOAP=\
# http://schemas.xmlsoap.org/wsdl/soap/
#

# maximum number of milliseconds to wait for a response to a synchronous request.
# Default value if not defined is to wait forever.
# Timeout properties are only used by providers that support timeouts.
wsif.syncrequest.timeout=10000

# maximum number of seconds to wait for a response to an async request.
# if not defined on invalid defaults to no timeout
# Timeout properties are only used by providers that support timeouts.
wsif.asyncrequest.timeout=60
```

To enable your existing web services to continue to work with WSIF, you might have to change the default WSIF SOAP provider back to the former Apache SOAP provider.

Chapter 32. Administering web services - Notification (WS-Notification)

WS-Notification enables web services to use the publish and subscribe messaging pattern.You use publish and subscribe messaging to publish one message to many subscribers. In this pattern a producing application inserts (publishes) a message (event notification) into the messaging system having marked it with a topic that indicates the subject area of the message. Consuming applications that have subscribed to the topic in question, and have appropriate authority, all receive an independent copy of the message that was published by the producing application.

Using WS-Notification for publish and subscribe messaging for web services

WS-Notification enables web services to use the publish and subscribe messaging pattern.

About this task

You use publish and subscribe messaging to publish one message to many subscribers. In this pattern a producing application inserts (publishes) a message (event notification) into the messaging system having marked it with a topic that indicates the subject area of the message. Consuming applications that have subscribed to the topic in question, and have appropriate authority, all receive an independent copy of the message that was published by the producing application.

Through the implementation of WS-Notification in WebSphere Application Server, you can achieve the following goals:

- Use existing service integration technologies and web services components to deliver WS-Notification functions.
- Interoperate with other publish and subscribe messaging clients (for example Java Message Service (JMS), WebSphere MQ) and with alternative message brokering products.

For more information about using WS-Notification, see the following topics:

Procedure

- “WS-Notification” on page 2964
- “Accomplishing common WS-Notification tasks” on page 2962
- “Configuring WS-Notification resources” on page 2985
- “Securing WS-Notification” on page 2970
- Developing applications that use WS-Notification

What to do next

Note:

- For development use on a single server only, you can use a script to configure the necessary resources to get up and running quickly with WS-Notification.
- In WebSphere Application Server Version 6.1, support for WS-Notification is based on a pre-final approval public review draft of the WS-Notification standards. In later versions, this support is extended to cover the final approved standards. The differences between the two versions of the standards are small, and your existing WS-Notification services and client applications will continue to work unchanged. However, you might choose to upgrade your existing applications as described in WebSphere Application Server Version 6.1 client applications must handle the additional error conditions introduced in the final Version 1.3 WS-Notification standards.

Accomplishing common WS-Notification tasks

This overview topic provides a set of links to information about the most common WS-Notification tasks. The tasks are divided into 3 groups by role: solution architect, system administrator and application developer.

About this task

Follow the links to view the detailed steps for each of the following tasks. For more information about the roles, see "WS-Notification roles and goals."

Procedure

- "Solution architect" on page 2963 sub-tasks:
  - "WS-Notification" on page 2964.
  - Selecting a hardware and software product combination for the enterprise that supports the WS-Notification standards.
  - Designing a server topology to host the applications, in accordance with particular WS-Notification topologies.
- "System administrator" on page 2963 sub-tasks:
  - "Using a script to get up and running quickly with WS-Notification" on page 2964.
  - "Configuring a WS-Notification service for use only by WS-Notification applications" on page 2968.
  - "Providing access for WS-Notification applications to an existing bus topic space" on page 2969.
  - "Securing WS-Notification" on page 2970.
  - "Configuring JAX-WS handlers" on page 2972.
  - "Applying a JAX-WS handler list to a WS-Notification service" on page 2973.
  - "Configuring a Version 7.0 WS-Notification service with Web service QoS" on page 2974.
  - "Configuring WS-Notification for reliable notification" on page 2976.
  - "Migrating a Version 6.1 WS-Notification configuration from WebSphere Application Server Version 6.1 to Version 7.0 or later" on page 2977.
  - "Preparing a migrated Version 6.1 WS-Notification configuration for reliable notification" on page 2978.
  - "Interacting at run time with WS-Notification" on page 2982.
  - "Publishing the WSDL files for a WS-Notification application to a compressed file" on page 2984.
- "Application developer" on page 2963 sub-tasks:
  - Developing applications that use WS-Notification
  - Writing a WS-Notification application that exposes a web service endpoint.
  - Writing a WS-Notification application that does not expose a web service endpoint.
  - Filtering the message content of publications.

WS-Notification roles and goals

This topics lists a set of computing roles that members of your organization might perform, and explains how you can use WS-Notification to help meet the goals of each role.

For a general description of each of the following roles, see WebSphere Application Server roles and goals.

Enterprise architect

IT environments are currently evolving towards the following concepts:

- Service Oriented Architecture (SOA)
- Enterprise Service Bus (ESB)

The goal of the enterprise architect might be to guide their organization towards appropriate utilization of these concepts to maximize the efficiency and responsiveness of the business as a whole.

WS-Notification enables publish and subscribe communication patterns (such as a stock ticker) to be exposed by using web services in an SOA environment. This is done through open standards, enabling straightforward replacement of the service implementation. It promotes easy exchange of data between suppliers and customers through use of standard web service operations and prevents vendor lock-in or adoption of proprietary standards.

WebSphere Application Server also allows WS-Notification to be used as an on- or off-ramp to an ESB, providing seamless interchange of data between different types of client connected to the bus.

Solution architect

The main goal of the solution architect is to design a solution that supports the specification set by the enterprise architect. This might include providing an environment in which web service applications can participate in publish and subscribe messaging patterns. This participation might also include the requirement to be able to exchange event notifications between web service clients and other clients of the enterprise service bus.

To create a design, the solution architect completes the following broad steps:

- Learn about the support provided for WS-Notification in WebSphere Application Server.
- Select a hardware and software product combination for the enterprise that supports the WS-Notification standards.
- Design a server topology to host the applications, in accordance with the particular WS-Notification topologies that are to be implemented.

System administrator

For the specific steps that the system administrator performs to help implement common WS-Notification tasks, see the following topics:

- “Using a script to get up and running quickly with WS-Notification” on page 2964.
- “Configuring a WS-Notification service for use only by WS-Notification applications” on page 2968.
- “Providing access for WS-Notification applications to an existing bus topic space” on page 2969.
- “Securing WS-Notification” on page 2970.
- “Configuring JAX-WS handlers” on page 2972.
- “Applying a JAX-WS handler list to a WS-Notification service” on page 2973.
- “Configuring a Version 7.0 WS-Notification service with Web service QoS” on page 2974.
- “Configuring WS-Notification for reliable notification” on page 2976.
- “Migrating a Version 6.1 WS-Notification configuration from WebSphere Application Server Version 6.1 to Version 7.0 or later” on page 2977.
- “Preparing a migrated Version 6.1 WS-Notification configuration for reliable notification” on page 2978.
- “Interacting at run time with WS-Notification” on page 2982.
- “Publishing the WSDL files for a WS-Notification application to a compressed file” on page 2984.

Application developer

If the solution architect specifies a requirement to insert event notifications into the system (that is publish messages) or receive event notifications from the system as a result of creating a subscription containing an interest profile, then the application developer can use WS-Notification to meet this requirement.

There are various patterns of producing and consuming application defined by WS-Notification that are available for use by the application developer, depending upon the exact requirements of the application in question. These options are explored in the following common WS-Notification tasks:

- Writing a WS-Notification application that exposes a web service endpoint.
- Writing a WS-Notification application that does not expose a web service endpoint.

See also Developing applications that use WS-Notification and Filtering the message content of publications.

WS-Notification

WS-Notification enables web services to use the publish and subscribe messaging pattern. Use these topics to learn more about WS-Notification.

You use publish and subscribe messaging to publish one message to many subscribers. In this pattern a producing application inserts (publishes) a message (event notification) into the messaging system having marked it with a topic that indicates the subject area of the message. Consuming applications that have subscribed to the topic in question, and have appropriate authority, all receive an independent copy of the message that was published by the producing application.

To learn about the WS-Notification implementation in WebSphere Application Server, see the following topics:

- ../ae/cjwsn_overview.dita
- WS-Notification: Benefits
- WS-Notification and end-to-end reliability
- WS-Notification terminology
- ../ae/cjwsn_interact.dita
- WS-Notification: Supported bindings
- WS-Notification and policy set configuration
- Reasons to create multiple WS-Notification services in a bus
- Reasons to create multiple WS-Notification service points
- Options for associating a permanent topic namespace with a bus topic space
- WS-Notification topologies

Using a script to get up and running quickly with WS-Notification

Use a jython script to configure the necessary resources to get up and running quickly with WS-Notification in WebSphere Application Server.

Before you begin

The example script provided in this topic is intended for development use on a single server only, and not for use in production or network deployment environments.

About this task

You can use the example script to configure a default set of resources that enable you to connect WS-Notification applications for development purposes. When executed, the script takes the following actions:

1. It searches in the configuration for an existing service integration bus, and creates one if necessary.
2. It searches for an existing bus member, and if one is not found it adds the (stand-alone) server to the bus, and uses the default data source.
3. It searches for an existing service integration bus topic space destination, and creates one if necessary.

4. It creates a Version 6.1 or Version 7.0 WS-Notification service.
5. It creates a Version 6.1 or Version 7.0 WS-Notification service point on the local server for a SOAP over HTTP binding.
6. It creates a WS-Notification permanent topic namespace to reference the service integration bus topic space that was found or created in step 3.
7. It saves the configuration, and describes where to find the WSDL for the new notification broker web service that has been exposed.

To use the example script, complete the following steps:

Procedure

1. Save the script to the file system and use a name of your choice (for example `wsnQuickStart.py`).
2. Modify the *hostRoot* variable defined at the top of the script to point to the HTTP port of the local server (usually 9080).
3. Install and configure the SDO repository.
4. Start the server, then execute the following command. If you saved the script with a name other than `wsnQuickStart.py`, then use that name instead.

   ```
   wsadmin -f wsnQuickStart.py
   ```

Example

Here is the example script:

```
#############################################################################
# WS-Notification QuickStart script                                         #
#                                                                           #
# This Jython script will quickly create the basic resources required in order to    #
# start using WS-Notification in WebSphere Application Server Version 6.1 or later   #
#                                                                           #
# Before executing it you must modify the variables defined below to match your      #
# configuration settings.                                                   #
#                                                                           #
# Note:                                                                     #
#    - This script is not intended for production use, and is intended for use on    #
#      a stand-alone server (not network deployment) only.                    #
#    - The script will search the configuration for an existing bus, and if one is   #
#      not found then a new bus will be created                             #
#    - It will then look for an existing Bus Member on the chosen bus. If one is not #
#      found then one will be created using the default File Store           #
#    - It will then look for an existing service integration bus topic space. If one #
#      is not found then it will create one.                                #
#                                                                           #
# Execute the script by typing;                                             #
#   wsadmin -f wsnQuickStart.py                                             #
#                                                                           #
#############################################################################

#########################################################
# Configuration variables                               #
#                                                       #
# Set the following variables to match your configuration #
#########################################################
# The URL root of HTTP port on the local server
hostRoot = "http://xyz.ibm.com:9080"
# The type of WS-Notification service you want to create (Version 6.1 or Version 7.0)
wsnServiceType = "V7.0"

#############################################################################
# Now create the configuration objects using the variables defined above    #
#############################################################################

# These variables are arbitrary choices and need not be set unless required.
wsnServiceName = "myWSNService"+wsnServiceType
wsnServicePointName = "myWSNServicePoint"+wsnServiceType
```

```
eplName = "myNewEPL"
tnsNamespaceURI = "http://example.org/topicNamespace/example1"

# General environment variables
nodeName = AdminTask.listNodes().split("\n")[0].rstrip()
server = AdminTask.listServers().split("\n")[0].rstrip()

print "#########################################################"
print "# Check the pre-requisites before you begin               #"
print "#########################################################"
# Check for the existence of the bus
requiresRestart = "false"
myBuses = AdminTask.listSIBuses().split("\n")
myBus = myBuses[0].rstrip()

if (myBus == ""):
        print "  *** Creating new bus "
        myBus = AdminTask.createSIBus("-bus MySampleBus -busSecurity false
                                       -scriptCompatibility 6.1")
        requiresRestart = "true"
#endIf
siBusName = AdminConfig.showAttribute(myBus, "name" )
print "  service integration bus name: "+siBusName+" "

# Check for the existence of the bus member
busMembers = AdminTask.listSIBusMembers(" -bus "+siBusName).split("\n")
myBusMember = busMembers[0].rstrip()

if (myBusMember == ""):
        print ""
        print "  *** Creating new Bus Member "
        busMemberName = AdminConfig.showAttribute(server, "name")
        myBusMember = AdminTask.addSIBusMember(" -bus "+siBusName+"
                          -node "+nodeName+" -server "+busMemberName)
        print ""
        requiresRestart = "true"
else:
        nodeName = AdminConfig.showAttribute(myBusMember, "node")
        busMemberName = AdminConfig.showAttribute(myBusMember, "server")
#endElse
print "  service integration bus Member on node: "+nodeName+" "
print "                                on server: "+busMemberName+" "

# Find a topic space to use
topicSpaces = AdminTask.listSIBDestinations(" -bus "+siBusName+"
                                  -type TopicSpace ").split("\n")
tSpace = topicSpaces[0].rstrip()

if (tSpace == ""):
        print "  *** Creating a Topic Space "
        tSpace = AdminTask.createSIBDestination(" -bus "+siBusName+"
                          -node "+nodeName+" -server "+myBusMember+"
                          -name MyTopicSpace -type TopicSpace")
        print ""

#endIf
siBusTopicSpaceName = AdminConfig.showAttribute(tSpace, "identifier" )
print "     service integration bus topic space: "+siBusTopicSpaceName+" "

print ""
print "#########################################################"
print "# Create the WS-Notification service                      #"
print "#########################################################"
newService = AdminTask.listWSNServices(" -name "+wsnServiceName+"
                             -bus "+siBusName).split("\n")[0].rstrip()

if (newService == ""):
        newService = AdminTask.createWSNService(" -name "+wsnServiceName+"
                             -bus "+siBusName+" "+" -type "+wsnServiceType)
        print "WS-Notification service created: "+wsnServiceName+" "
        print "                             on bus: "+siBusName+" "
else:
```

```
        print "WS-Notification service '"+wsnServiceName+"' "
        print "  already exists on bus '"+siBusName+"' "
#endElse

print ""
print "#########################################################"
print "# Create the WS-Notification service point                #"
print "#########################################################"
eplURLRoot = hostRoot+"/wsn"
wsdlURLRoot = hostRoot+"/SIBWS/wsdl"

newServicePoint = AdminTask.listWSNServicePoints(newService, "
                -name "+wsnServicePointName+" " ).split("\n")[0].rstrip()

if (newServicePoint == ""):
        if (wsnServiceType == "V7.0"):
            newServicePoint = AdminTask.createWSNServicePoint(newService, "
            -name "+wsnServicePointName+" -node "+nodeName+" -server "+busMemberName)
        else:
            newServicePoint = AdminTask.createWSNServicePoint(newService, "
            -name "+wsnServicePointName+"
            -node "+nodeName+" -server "+busMemberName+" -eplName "+eplName+"
            -eplURLRoot "+eplURLRoot+" -eplWSDLServingURLRoot "+wsdlURLRoot+" " )
        print "WS-Notification service point created: "+wsnServicePointName+" "
        print "                         on bus member: "+busMemberName+" "
        print "                               on node: "+nodeName+" "
else:
        print "WS-Notification service point '"+wsnServicePointName+"' "
        print "already exists on WS-Notification service '"+wsnServiceName+"' "
#endElse

print ""
print "#########################################################"
print "# Create the WS-Notification permanent topic namespace    #"
print "#########################################################"
newTopicNamespace = AdminTask.listWSNTopicNamespaces(newService, "
                    -namespace "+tnsNamespaceURI+" ").split("\n")[0].rstrip()

if (newTopicNamespace == ""):
    newTopicNamespace = AdminTask.createWSNTopicNamespace(newService, "
      -namespace "+tnsNamespaceURI+" -busTopicSpace "+siBusTopicSpaceName+"
      -reliability RELIABLE_PERSISTENT")
  print "WS-Notification topic namespace created: "+tnsNamespaceURI+" "
  print "                         bus topic space: "+siBusTopicSpaceName+" "
else:
  print "WS-Notification permanent topic namespace already exists: "
        +tnsNamespaceURI+" "
#endElse

#############################################################################
# All the objects have been created - inform the user where to proceed next #
#############################################################################

print ""
print "#########################################################"
print "# Summary                                               #"
print "#########################################################"

# Calculate where you would find the WSDL for the new service.
if (wsnServiceType == "V7.0"):
  print "IMPORTANT: Because you've created a Version 7.0 service"
  print "you need to start the newly installed application;"
  print "           WSN_"+wsnServiceName+"_"+wsnServicePointName
  print ""
  wsdlLocation = hostRoot+"/"+wsnServiceName+wsnServicePointName
                       +"NB/NotificationBroker?wsdl"
else:
  print "IMPORTANT: Because you've created a Version 6.1 service"
  print "you need to start 2 newly installed applications;"
  serverName = AdminConfig.showAttribute(server, "name")
  print "           "+eplName+"."+nodeName+"."+serverName
  print "           sibws."+nodeName+"."+serverName
```

```
  print ""
  wsdlLocation = hostRoot+"/wsn/soaphttpengine/"+siBusName+"/"
                 +wsnServiceName+"NotificationBroker/"
                 +wsnServicePointName+"NotificationBrokerPort?wsdl"
print " The WSDL for the new service can be viewed at the following location; "
print "   "+wsdlLocation+" "
print ""
print " Your web service applications can publish and subscribe to any topics in the namespace; "
print "   "+tnsNamespaceURI+" "
print ""
if (requiresRestart == "true"):
  print " You must now restart the server for the changes to take effect. "
  print ""
#endIf

print ""
print "#########################################################"
print "# Save the configuration and exit                       #"
print "#########################################################"
AdminConfig.save()
sys.exit()
```

Configuring a WS-Notification service for use only by WS-Notification applications

Define all the necessary objects to configure a WS-Notification service and associated service points.

Before you begin

This task assumes that you have already configured one or more application servers, and that there are no other messaging resources configured.

For Version 6.1 WS-Notification services, you must also create an endpoint listener configuration on each application server that you want to use to host a WS-Notification service.

About this task

You can configure one or more servers, each hosting a WS-Notification service to which web services applications can connect, as described in the Simple web services topology.

Procedure

1. Use the information given in “Creating a bus” on page 1884 to create a service integration bus and add your existing server or servers as bus members. When you add a bus member, a messaging engine is created in the server. You choose the database that hosts the persistent storage required by the messaging engine. You can either use the default JDBC data source and Apache Derby JDBC provider, or you can configure a data source for use with your preferred database product.
2. Using the administrative console, navigate to **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The “WS-Notification services [Collection]” on page 3047 form for this bus is displayed.
3. Click **New**. The New WS-Notification service wizard is displayed.
4. Complete steps 1 to 4 of the wizard, as described in “Creating a new Version 6.1 WS-Notification service” on page 2995 or “Creating a new Version 7.0 WS-Notification service” on page 2987. Pay particular attention to the following settings:
   - In wizard step 1: “Configure name, description, service integration bus and dynamic topic namespace settings”, you select the service integration bus to use to host the messaging resources. Choose the bus that you created at the beginning of this task.
   - In wizard step 4: “Create WS-Notification service points”, you select the bus member that is to host the new service. On a single server system there is only one option here. For Version 6.1 WS-Notification services, you also select the endpoint listener application to use to expose the service. Choose the endpoint listener application that you configured before you began this task.

5. Optional: In wizard step 5: “Create permanent topic namespaces”, configure the WS-Notification topic namespace to provide access to a service integration bus topic space:
   - Enter the topic namespace URI that you want WS-Notification applications to use when referring to the service integration bus topic space. This must be unique within the WS-Notification service, and is conventionally a URI related to your organization. For example `http://www.myorganization.com`.
   - Because there are no existing service integration bus topic spaces in your bus the “Use an existing service integration bus topic space” radio button and associated drop-down list are not available. To create a new bus topic space, enter your chosen name.
6. Complete wizard step 6: “Summary”.

   Check that the summary of the actions taken by the wizard is as you expected, then click **Finish**. If the processing completes successfully, the list of WS-Notification services is updated to include the new WS-Notification service. Otherwise, an error message is displayed.
7. Save your changes to the master configuration. You do not have to restart the server for these changes to take effect. However, you must start the endpoint listener or enterprise application associated with the service point that was created in step 4 of the wizard.

What to do next

For JAX-WS based Version 7.0 WS-Notification services, you can view the URL to which WS-Notification applications connect by looking in the `NotificationBroker.wsdl` file for the NotificationBroker application. To view this file, see “Publishing the WSDL files for a WS-Notification application to a compressed file” on page 2984.

For JAX-RPC based Version 6.1 WS-Notification services, you can view the URL to which WS-Notification applications connect by navigating to **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> NotificationBroker inbound port settings**.

To extend this configuration so that web service applications can interact with non-web service applications, see “Providing access for WS-Notification applications to an existing bus topic space.”

Providing access for WS-Notification applications to an existing bus topic space

Configure a solution such that web services clients can share event notifications with other clients of a service integration bus.

Before you begin

This task assumes that you have an existing service integration bus, configured with at least one bus member and a bus topic space. For more information, see “Creating a bus” on page 1884.

About this task

This task focuses on the “Create permanent topic namespaces” step that is part of the New WS-Notification service wizard.

You can configure WS-Notification so that web service applications receive event notifications generated by other clients of the service integration bus such as JMS clients. Similarly, web service applications can generate notifications to be received by other client types. This configuration is described in the Topology for WS-Notification as an entry or exit point to the service integration bus. You achieve this configuration by creating a permanent topic namespace that allows messages to be shared between web service and non-web service clients of the bus. Specifically, you create a permanent topic namespace that links the service integration bus topic space used by messaging clients to a WS-Notification topic namespace URI.

For more information about programming the client applications, see Interacting with JMS message types.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The “WS-Notification services [Collection]” on page 3047 form is displayed.
3. Click **New**. The New WS-Notification service wizard is displayed.
4. Complete steps 1 to 4 of the wizard, as described in “Creating a new Version 6.1 WS-Notification service” on page 2995 or “Creating a new Version 7.0 WS-Notification service” on page 2987.
5. In wizard step 5:“ Create permanent topic namespaces”, configure the WS-Notification topic namespace to provide access to the existing service integration bus topic space:
   - Enter the topic namespace URI that you want WS-Notification applications to use when referring to the service integration bus topic space. This must be unique within the WS-Notification service, and is conventionally a URI related to your organization. For example `http://www.myorganization.com`.
   - Select the “Use an existing service integration bus topic space” radio button, then from the drop-down list select the name of your chosen service integration bus topic space.
6. Complete wizard step 6: “Summary”.

   Check that the summary of the actions taken by the wizard is as you expected, then click **Finish**. If the processing completes successfully, the list of WS-Notification services is updated to include the new WS-Notification service. Otherwise, an error message is displayed.
7. Save your changes to the master configuration. You do not have to restart the server for these changes to take effect. However, you must start the endpoint listener or enterprise application associated with the service point that was created in step 4 of the wizard.

Results

WS-Notification applications can now connect to the WS-Notification service point and insert or receive event notifications from the service integration bus topic space.

What to do next

For JAX-WS based Version 7.0 WS-Notification services, you can view the URL to which WS-Notification applications connect by looking in the `NotificationBroker.wsdl` file for the NotificationBroker application. To view this file, see “Publishing the WSDL files for a WS-Notification application to a compressed file” on page 2984.

For JAX-RPC based Version 6.1 WS-Notification services, you can view the URL to which WS-Notification applications connect by navigating to **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> NotificationBroker inbound port settings**.

Securing WS-Notification

The WS-Notification security implementation requires that a user identity is flowed in requests for WS-Notification services. This identity is used to authenticate the client application and check that the client is authorized to invoke the requested operation, and to access the underlying service integration bus topic spaces and topic resources.

About this task

WS-Notification uses the same mechanisms as other Web services to provide an authenticated identity. For example WS-Security or HTTP Basic Authentication.

There are three parts to configuring secure access to WS-Notification:

- Securing the communication channel between the application and the server.

- Authorizing the application to invoke the NotificationBroker.
- Authorizing the application to access the resources of the service integration bus.

If messaging security is enabled, and the WS-Security or HTTP Basic Authentication components are not configured to flow a user identity in WS-Notification requests, then all such requests are treated as unauthenticated and can only access messaging resources that are accessible by the WebSphere Application Server “everyone” group.

Procedure

1. Secure the communication between the application and the server:
   a. Provide security for inbound requests and associated responses by configuring the WS-Notification service point.
      - For JAX-WS based Version 7.0 WS-Notification service points, attach security-enabled policy sets to the application associated with the service point. For JAX-RPC based Version 6.1 WS-Notification service points, configure security for the inbound ports associated with the service point.
      - If you are using SOAP over HTTP as the binding for your WS-Notification service point, modify it to use SOAP over HTTPS as described in Configuring secure access to WS-Notification service points by using SOAP over HTTPS.
      - If you are using SOAP over JMS as the binding (for Version 6.1 WS-Notification services), configure the JMS connection factory used by the client application to use a secure communication protocol to communicate with the JMS provider. Exactly how you do this depends upon the JMS provider. If you are using the service integration bus as the JMS provider, configure the client to use SSL to communicate with the server by setting the **target inbound transport chain** to `InboundSecureMessaging` as described in How JMS applications connect to a messaging engine on a bus and its related tasks.
   b. Provide security for outbound requests (for example notifications from the server to subscribed consumers) by configuring the WS-Notification service.
      - For JAX-WS based Version 7.0 WS-Notification services, the steps involved are similar to those for applying security to JAX-WS web service clients except that any binding or configuration created is applied to the WS-Notification service. For more information, see Securing JAX-WS web services using message-level security.
      - For JAX-RPC based Version 6.1 WS-Notification services, the steps involved are similar to those for applying security to service integration bus-enabled web services outbound ports except that any binding or configuration created is applied to the WS-Notification service. For more information, see Securing bus-enabled web services and its sub-topics, notably Invoking outbound services over HTTPS.
   c. You can also use WS-Security to sign or encrypt SOAP messages.
      - For JAX-WS based Version 7.0 WS-Notification services, see Signing and encrypting message parts using policy sets.
      - For JAX-RPC based Version 6.1 WS-Notification services, see Configuring secure transmission of SOAP messages by using WS-Security.
2. Authorize the application to invoke the NotificationBroker:
   a. Configure the client application to provide an appropriate identity.

      To authorize a web service application to communicate with the server, the application must identify itself as running as a particular authenticated identity. The mechanism for doing this depends upon the type of web service binding you are using:
      - If you are using SOAP over HTTP web service bindings, use either HTTP Basic Authentication or WS-Security to provide the authenticated identity.
      - If you are using SOAP over JMS web service bindings (for Version 6.1 WS-Notification services), use WS-Security to provide an authenticated identity.

b. Configure the server to authorize the client application identity to carry out the required operations.
   - For JAX-WS based Version 7.0 WS-Notification services, you can use Web services policy sets such as the “Username WS-I RSP default” or “Username WSSecurity default” policy sets to apply authorization to the Web services that are deployed in the enterprise application associated with a service point. See also the IBMdeveloperWorks article Configuring JAX-WS applications with WS-Security for WS-Notification.
   - For JAX-RPC based Version 6.1 WS-Notification services, you can apply authorization to the whole of an inbound service (for example the NotificationBroker endpoint of a WS-Notification service point) as described in Password-protecting inbound services, or configure authorization constraints independently for each Web service operation as described in Password-protecting a web service operation.

3. Authorize the application to access the resources of the service integration bus.

   Service integration bus security uses role-based authorization. When a user is assigned to a role, the user is granted all of the permissions that the role contains. By administering authorization permissions, you can control user access to a bus and to its resources when messaging security is enabled.

   a. Authorize the application identity to be able to connect to the service integration bus, as described in Administering the bus connector role.
   b. When the application can connect to the bus, grant the application access to the appropriate destinations on the bus.

      You can determine which service integration bus topic spaces are required, by checking which WS-Notification topic namespaces are used by the application then looking at the appropriate WS-Notification permanent topic namespace to find the service integration bus topic space to which it maps. You can then grant authorization (for example the Sender or Receiver roles) for the authenticated identity to access that topic space as described in “Administering destination roles” on page 1991.
   c. After the client application has been authorized to access the appropriate topic space destination, you might also need to authorize the client application to access the individual topics within the topic space destination as described in Administering topic roles.

   For general information about configuring access to the service integration bus, see Securing service integration.

Configuring JAX-WS handlers

A JAX-WS handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. You can create JAX-WS handlers, chain them together in the form of a handler list, then apply the handler list to a JAX-WS based Version 7.0 WS-Notification service point (for inbound invocation handling) or WS-Notification service (for outbound invocation handling).

About this task

The Java API for XML-based Web Services (JAX-WS) provides you with a standard way of developing interoperable and portable web services. To create a JAX-WS handler, you can use a tool such as IBM Rational Application Developer. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more JAX-WS based Version 7.0 WS-Notification services or service points, so that the handler list can monitor WS-Notification activity and take appropriate action depending upon the sender and content of each inbound or outbound message.

Detailed instructions on how to configure JAX-WS handlers and handler lists for use with JAX-WS based Version 7.0 WS-Notification services are provided in the following topics:

Procedure

- Load JAX-WS handler classes.
- Create a new JAX-WS handler configuration.
- Modify an existing JAX-WS handler configuration.
- Delete JAX-WS handler configurations.
- Create a new JAX-WS handler list.
- Modify an existing JAX-WS handler list.
- Delete JAX-WS handler lists.

Applying a JAX-WS handler list to a WS-Notification service

To handle the messages that flow to and from an existing JAX-WS based Version 7.0 WS-Notification service, you must create JAX-WS handlers, chain them together in the form of a handler list, then apply the handler list to a NotificationBroker, PublisherRegistrationManager or SubscriptionManager endpoint at a Version 7.0 WS-Notification service point (for inbound invocation handling), or apply the handler list to a WS-Notification service (for outbound invocation handling).

Before you begin

This task assumes that you have already created a Version 7.0 WS-Notification service.

About this task

To create a JAX-WS handler, you can use a tool such as IBM Rational Application Developer. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more JAX-WS based Version 7.0 WS-Notification services or service points, so that the handler list can monitor WS-Notification activity and take appropriate action depending upon the sender and content of each inbound or outbound message. For example:

- You can use a handler list on the NotificationBroker web service to log all notification messages received by this service point.
- You can use a handler list on a SubscriptionManager web service to log all unsubscribe requests received by this service point.
- You can use a handler list on a PublisherRegistrationManager web service to log all publisher deregistration requests received by this service point.

Procedure

1. Create one or more JAX-WS handlers. You can do this using IBM Rational Application Developer or a similar tool.
2. Load JAX-WS handler classes. A JAX-WS handler interacts with messages through a JAX-WS based Version 7.0 WS-Notification service point (for inbound invocation handling) or WS-Notification service (for outbound invocation handling), therefore you must make the handler class available to the server or cluster that hosts the WS-Notification service point or service that you want to monitor.
3. Create a new JAX-WS handler configuration by using the administrative console or by using the “createJAXWSHandler command” on page 3086. By creating a new handler configuration, you make WebSphere Application Server aware of your handler, and you make the handler available for inclusion in one or more handler lists.
4. Create a new JAX-WS handler list. The approach taken in WebSphere Application Server is to assign handler lists (rather than individual handlers) to WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling).
5. Optional: To apply a JAX-WS handler list to a service provider endpoint (NotificationBroker, PublisherRegistrationManager or SubscriptionManager) associated with a service point, use the administrative console to complete the following substeps:

a. Navigate to **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**. The "WS-Notification service points [Collection]" on page 3043 form is displayed. This form shows all the service points configured for this Version 7.0 WS-Notification service.

b. In the content pane, click the name of a JAX-WS based Version 7.0 WS-Notification service point in the list. The current settings for this Version 7.0 WS-Notification service point are displayed in the "WS-Notification service points [Settings]" on page 3043 form.

c. Apply the JAX-WS handler list by selecting it from the list box for one or more of the following general properties:

NotificationBroker JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the NotificationBroker endpoint of the WS-Notification service point.

SubscriptionManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the SubscriptionManager endpoint of the WS-Notification service point.

PublisherRegistrationManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the PublisherRegistrationManager endpoint of the WS-Notification service point.

6. Optional: To apply a JAX-WS handler list to a WS-Notification service, use the administrative console to complete the following substeps:

a. Navigate to **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The "WS-Notification services [Collection]" on page 3047 form is displayed.

b. In the content pane, click the name of a JAX-WS based Version 7.0 WS-Notification service in the list. The current settings for this Version 7.0 WS-Notification service are displayed in the "WS-Notification services [Settings]" on page 3047 panel.

c. Apply the JAX-WS handler list by selecting it from the list box for the following general property:

JAX-WS handler list

The JAX-WS handler list that is applied to outbound requests from the WS-Notification service.

Configuring a Version 7.0 WS-Notification service with Web service QoS

You use the administrative console to configure web service qualities of service (QoS) such as reliability or security by applying policy sets to a Version 7.0 WS-Notification service. The configuration of policy sets for WS-Notification can be split into two types: service provider (for Version 7.0 WS-Notification service points) and service client (for Version 7.0 WS-Notification service clients). Policy set configuration for these two elements of a WS-Notification service is handled differently.

Before you begin

This task assumes that you have already created a Version 7.0 WS-Notification service. You must also have configured policy sets that meet your quality of service requirements. You can reuse existing policy sets within your organization, use the default policy sets provided by WebSphere Application Server, or create new policy sets. For more information, see "Managing policy sets using the administrative console" on page 2666.

About this task

Version 7.0 WS-Notification service points (NotificationBrokers, PublisherRegistrationManagers and SubscriptionManagers) are implemented as JAX-WS applications that are deployed to servers or clusters, and the management of the policy sets for these WS-Notification service points is configured by using the policy set administrative infrastructure for service providers (both panels and wsadmin commands). In the Service provider [settings] panel for a WS-Notification service provider, there is a link to the associated WS-Notification service point application. You can use the service provider settings panel to attach policy sets to each NotificationBroker, PublisherRegistrationManager and SubscriptionManager.

Version 7.0 WS-Notification services are implemented as two configurable service clients (OutboundNotificationService and OutboundRemotePublisherService) for each WS-Notification service. Events that need outgoing web service invocations occur at the service integration bus level rather than the bus member level, even though notifications of the events occur within a particular bus member. The two service clients that a WS-Notification service implements are therefore configured by using the policy set administrative infrastructure for service clients. In the Service client policy sets and bindings [collection] panel, the two service clients for a given WS-Notification service are listed in a tree structure, along with their endpoints and operations. You can use this panel to attach policy sets to each service client, or to both service clients for the WS-Notification service.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The “WS-Notification services [Collection]” on page 3047 form is displayed.
3. In the content pane, click the name of the Version 7.0 WS-Notification service to which to apply web service qualities of service. The current settings for this Version 7.0 WS-Notification service are displayed in the “WS-Notification services [Settings]” on page 3047 panel.
4. Optional: To configure web service QoS for inbound requests, apply policy sets to the service provider applications (NotificationBroker, PublisherRegistrationManager and SubscriptionManager) associated with a service point:

   Click **[Additional Properties] WS-Notification service points**. The “WS-Notification service points [Collection]” on page 3043 form is displayed. This form shows all the service points configured for this Version 7.0 WS-Notification service.

   Repeat the following steps, as required, to configure the policy set and binding information for one or more service provider applications for one or more service points:

   a. In the content pane, click the name of a Version 7.0 WS-Notification service point in the list. The current settings for this Version 7.0 WS-Notification service point are displayed in the “WS-Notification service points [Settings]” on page 3043 form.

   b. Click **[Additional Properties] Policy set configuration.** A link to each of the service provider applications (NotificationBroker, PublisherRegistrationManager and SubscriptionManager) for this service point is displayed in the Service provider [Collection] form.

   c. Click the name of one of the applications. An indented list of endpoints and operations for this service is displayed in the Service provider [settings] form.

   d. Select one or more items in the list (service, endpoint or operations), then perform one of the three available operations on the items you have selected:

      - Attach policy set
      - Detach policy set
      - Assign binding

      For more information, see “Managing policy sets using the administrative console” on page 2666.

   e. Restart the service provider application.

5. Optional: To configure web service QoS for outbound requests, apply policy sets to one or both of the web service clients associated with the WS-Notification service:
   a. If necessary, navigate back to the "WS-Notification services [Settings]" on page 3047 panel for this Version 7.0 WS-Notification service.
   b. Click **[Additional Properties] Outbound request policy sets and bindings**. The two service clients for this WS-Notification service are listed in a tree structure, along with their endpoints and operations, in the Service client policy set and bindings [collection] form.
   c. Select one or both service clients, then perform one of the three available operations on the items you have selected:
      - Attach client policy set
      - Detach client policy set
      - Assign binding

      For WS-Notification service clients, policy set attachments are not supported at the endpoint (port) or operation level. Therefore endpoints or operations are not selectable, and are shown as inheriting any policy set or binding that is attached to the service client. For more information, see "Managing policy sets using the administrative console" on page 2666.

      Note: Using the Service client policy set and bindings [collection] form, you can configure the policy set and binding information for both service clients for the selected service. Alternatively, you can configure the policy set and binding information for a single Version 7.0 WS-Notification service client by clicking **Services -> Service clients -> *ws-notification_service_client_name*** and using the WS-Notification service client [settings] panel. This panel also provides links to the associated service integration bus and WS-Notification service.

Results

The WS-Notification service has been successfully enabled with the required web service QoS.

Configuring WS-Notification for reliable notification

To ensure that WS-Notification web service interactions are performed in a reliable way, you configure a JAX-WS based Version 7.0 WS-Notification service, JAX-WS client, and JAX-WS based WS-Notification consumer web service (through policy set functions) to use WS-ReliableMessaging.

Before you begin

This task assumes that you have created a Version 7.0 WS-Notification service and service point, and that you have created two JAX-WS applications:

- A JAX-WS client application, created from the WSDL of the new service point.
- A JAX-WS based WS-Notification consumer web service.

For more information about how to create these applications, see "Using JAX-WS clients and web services with new Version 7.0 WS-Notification service points" on page 2981.

About this task

Reliable notification refers to the reliable transmission of messages to and from the IBM WS-Notification implementation. You enable this reliability to mitigate the problems inherent in network transmission protocols such as HTTP.

Note: This reliability does not include the behavior of the application server itself. For more information, see WS-Notification and end-to-end reliability.

To enable reliable notification, you apply policy sets that include the WS-ReliableMessaging policy to the service point, service client, and service consumer applications.

You can configure policy sets for JAX-WS clients for both application server and client environments, including thin clients. For more information, see “Managing policy sets using the administrative console” on page 2666.

Procedure

1. Configure policy sets that meet your reliable messaging requirements.

   You can reuse existing policy sets within your organization, use the WS-ReliableMessaging default policy sets provided by WebSphere Application Server, or create new policy sets. For more information, see “Configuring a WS-ReliableMessaging policy set by using the administrative console” on page 3117.
2. Configure the Version 7.0 WS-Notification service and service points for reliable notification.

   Apply policy sets that include the WS-ReliableMessaging policy.
3. Configure the JAX-WS based WS-Notification client application so that it interacts reliably with its target web service by using WS-ReliableMessaging.

   Apply policy sets that include the WS-ReliableMessaging policy, as described in “Configuring a Version 7.0 WS-Notification service with Web service QoS” on page 2974 and “Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the administrative console” on page 3122.
4. Configure the JAX-WS based WS-Notification consumer web service application so that it interacts reliably with clients that attempt to communicate with it.

   Apply policy sets that include the WS-ReliableMessaging policy, as described in “Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the administrative console” on page 3122.
5. Configure the WS-Notification client application (which has been configured to interact reliably) to communicate with the WS-Notification service point (which has similarly been configured to receive messages reliably).
6. Prompt the WS-Notification client application to subscribe on behalf on the (reliably configured) WS-Notification consumer web service.

   Note: The WS-Notification consumer web service might also act as a client to perform its own subscription. This client would require additional policy set configuration if reliable interactions were required.
7. Initiate notification message publications from the WS-Notification client application.

Results

The JAX-WS based Version 7.0 WS-Notification service point receives the notification messages in a reliable way from the WS-Notification client, and publishes the notification messages in a reliable way to the WS-Notification consumer web service.

Migrating a Version 6.1 WS-Notification configuration from WebSphere Application Server Version 6.1 to Version 7.0 or later

You can migrate an existing Version 6.1 WS-Notification configuration to run in a WebSphere Application Server Version 7.0 or later environment.

Before you begin

This topic assumes that you have the following configuration:

- An existing server or cluster installation of WebSphere Application Server Version 6.1 that is ready for migration, including at least one WS-Notification service, service point, and underlying service integration bus.
- Existing subscriptions with associated messages pending notification.
- Existing operational WS-Notification client and web service applications that are known to transmit and receive notifications through the existing Version 6.1 WS-Notification service.

About this task

When you migrate your system from WebSphere Application Server Version 6.1 to Version 7.0 or later, any existing WS-Notification configuration (for example WS-Notification services, service points, administered subscribers), and the associated service integration bus resources (for example inbound services and endpoint listeners) are automatically migrated to valid configuration elements within the new installation. The services and service points are migrated to Version 6.1 WS-Notification services and service points, the endpoint URLs for the service points are unchanged, and the functions supported are exactly the same as in WebSphere Application Server Version 6.1. For example, reliable notification is not possible with Version 6.1 WS-Notification services and service points.

Any messages pending notification are delivered following the update and server startup sequence.

Procedure

1. Migrate your product configuration from WebSphere Application Server Version 6.1 to Version 7.0 or later.
2. Check that notifications are being published and consumed correctly in your migrated system.

   Initiate the program, process, or application that includes the WS-Notification client (or clients), and that causes notifications to be requested and transmitted.

Results

The WS-Notification services and resources are migrated to Version 6.1 WS-Notification services and resources that run under WebSphere Application Server Version 7.0 or later. Any messages pending notification are delivered.

What to do next

You are now ready (optionally) to prepare your migrated Version 6.1 WS-Notification configuration for reliable notification.

Preparing a migrated Version 6.1 WS-Notification configuration for reliable notification

You can gradually introduce JAX-WS based client and provider entities such that a migrated Version 6.1 WS-Notification configuration is ready to be configured for reliable notification.

Before you begin

This topic assumes that you have an existing server or cluster installation of WebSphere Application Server Version 7.0 or later, including at least one WS-Notification service, service point, and underlying service integration bus that has been migrated to this version of the product as described in “Migrating a Version 6.1 WS-Notification configuration from WebSphere Application Server Version 6.1 to Version 7.0 or later” on page 2977.

About this task

For reliable notification, you apply policy sets that include WS-ReliableMessaging to your WS-Notification configuration. You can only use policy sets with Java API for XML-based Web Services (JAX-WS) applications, and with Version 7.0 WS-Notification services and service points.

The WS-Notification implementation in WebSphere Application Server Version 6.1 uses service integration bus-enabled web services to expose the WS-Notification service endpoint, so that it can be invoked by applications and configured with specific attributes such as WS-Security or JAX-RPC handlers. However, the Version 6.1 implementation is not compatible with JAX-WS handlers or applications, and it cannot compose with WS-ReliableMessaging.

To prepare a migrated Version 6.1 WS-Notification configuration for reliable notification, you must recreate your Version 6.1 WS-Notification services and service points as Version 7.0 WS-Notification services and service points, and recreate each JAX-RPC client application to which you want to apply a policy set as a JAX-WS application. Note that you can continue to use JAX-RPC applications with Version 7.0 WS-Notification services and service points, and that you only have to recreate those applications that must work with policy sets.

For information about coding JAX-RPC and JAX-WS client applications to perform specific WS-Notification tasks, see ../ae/tjwsn_devapp.dita. You might also find it useful to learn about JAX-WS and the JAX-WS client programming model. This should help you to determine the effort involved in porting client code from JAX-RPC to JAX-WS, or to validate JAX-WS client to JAX-RPC web service interoperability.

To support a phased approach to preparing for reliable notification, and to describe the four main configurations that you might want to achieve, this task is divided into four subtasks:

Procedure

- "Using JAX-WS clients and web services with migrated service points."
- "Using JAX-RPC clients and web services with new Version 7.0 WS-Notification service points" on page 2980.
- "Using JAX-WS clients and web services with new Version 7.0 WS-Notification service points" on page 2981.
- "Sharing notifications between Version 6.1 and Version 7.0 WS-Notification service points" on page 2982.

What to do next

When you have completed these subtasks, you have a collection of WS-Notification client and server entities that are prepared for reliable notification, and you are ready to configure WS-Notification for reliable notification.

Using JAX-WS clients and web services with migrated service points:
Procedure

1. Publish notification messages through a migrated Version 6.1 WS-Notification service point, from a JAX-WS client application.
   a. Create a JAX-WS WS-Notification client application by using the WSDL of the migrated service point.

      For more information, see Example: Publishing a WS-Notification message, Developing a JAX-WS client from a WSDL file and "Publishing the WSDL files for a WS-Notification application to a compressed file" on page 2984.
   b. Run the application.
   c. Initiate one or more notification messages.

The system accepts and publishes the notification messages from the JAX-WS client.

2. Receive notification messages in a new JAX-WS based WS-Notification consumer application, from a migrated Version 6.1 WS-Notification service point.

   This validates that your Version 6.1 WS-Notification service point can deliver notifications to a JAX-WS consumer web service.

   a. Create a new JAX-WS based WS-Notification consumer web service from the standard WS-Notification WSDL.

      For more information, see Example: Subscribing a WS-Notification consumer, Implementing web services applications from existing WSDL files with JAX-WS and "Publishing the WSDL files for a WS-Notification application to a compressed file" on page 2984.

   b. Create a subscription for the new consumer service through the Version 6.1 WS-Notification service point.

   c. Prompt the WS-Notification service point to generate notifications (for example by using a WS-Notification client application).

   The system transmits the notifications to the new JAX-WS consumer application correctly.

Using JAX-RPC clients and web services with new Version 7.0 WS-Notification service points:
Procedure

1. Create a new Version 7.0 WS-Notification service.

   You can configure a Version 7.0 WS-Notification service and service points with policy sets to compose with WS-ReliableMessaging for reliable notification. The system creates and configures a new Version 7.0 WS-Notification service. This includes the creation of a Version 7.0 WS-Notification service point that exposes the service from a particular service integration bus member. Version 6.1 and Version 7.0 WS-Notification service points can coexist in WebSphere Application Server Version 7.0 or later.

2. Publish notification messages through the new Version 7.0 WS-Notification service point, from a JAX-RPC client application.

   This validates the behavior of the Version 7.0 WS-Notification service point.

   a. Create the application by using the WSDL of the new Version 7.0 WS-Notification service point.

      For more information, see Example: Publishing a WS-Notification message, Developing client bindings from a WSDL file for a JAX-RPC Web services client and "Publishing the WSDL files for a WS-Notification application to a compressed file" on page 2984.

      Note: Instead of creating a new JAX-RPC client application, you might choose to update an existing JAX-RPC client application from the WSDL of the new service point. The WSDL for a Version 7.0 WS-Notification service point contains a number of minor changes compared to a Version 6.1 service point, so you must modify your existing JAX-WS client application to take account of these changes. Specifically, you must regenerate the java proxy classes from the WSDL, and update any use of class names and methods that have changed. For example, there might be changes in the generated classes that include a port type or service from the WSDL.

   b. Run the application.

   c. Initiate one or more notification messages.

   The system accepts and publishes the notification messages from the JAX-RPC client.

3. Receive notification messages in a JAX-RPC based WS-Notification consumer application, from the new Version 7.0 WS-Notification service point.

   This validates that your Version 7.0 WS-Notification service point can deliver notifications to a JAX-RPC consumer web service.

   a. Create a new JAX-RPC based WS-Notification consumer web service from the standard WS-Notification WSDL.

For more information, see Example: Subscribing a WS-Notification consumer, Implementing web services applications from existing WSDL files with JAX-WS and “Publishing the WSDL files for a WS-Notification application to a compressed file” on page 2984.

Note: Instead of creating a new JAX-RPC consumer application, you can use an existing JAX-RPC consumer application from (for example) a Version 6.1 WS-Notification configuration.

b. Create a subscription for the new consumer service through the new Version 7.0 WS-Notification service point.

c. Prompt the WS-Notification service point to generate notifications (for example by using a WS-Notification client application).

The system transmits the notifications to the new JAX-RPC consumer application correctly.

Using JAX-WS clients and web services with new Version 7.0 WS-Notification service points:
Before you begin

Note that with this configuration you can compose with policy sets for reliable notification.

Procedure

1. Publish notification messages through the new Version 7.0 WS-Notification service point, from a JAX-WS client application.

   a. Create a JAX-WS WS-Notification client application by using the WSDL of the new Version 7.0 WS-Notification service point.

   For more information, see Example: Publishing a WS-Notification message, Developing a JAX-WS client from a WSDL file and “Publishing the WSDL files for a WS-Notification application to a compressed file” on page 2984.

   Note: Instead of creating a new JAX-WS client application, you might choose to update the JAX-WS client application that you created in the subtask “Using JAX-WS clients and web services with migrated service points” on page 2979. The WSDL for a Version 7.0 WS-Notification service point contains a number of minor changes compared to a Version 6.1 service point, so you must modify your existing JAX-WS client application to take account of these changes. Specifically, you must regenerate the java proxy classes from the WSDL, and update any use of class names and methods that have changed. For example, there might be changes in the generated classes that include a port type or service from the WSDL.

   b. Run the application.

   c. Initiate one or more notification messages.

   The system accepts and publishes the notification messages from the JAX-WS client.

2. Receive notification messages in a new JAX-WS based WS-Notification consumer application, from a new Version 7.0 WS-Notification service point.

   This validates that your Version 7.0 WS-Notification service point can deliver notifications to a JAX-WS consumer web service.

   a. Create a new JAX-WS based WS-Notification consumer web service from the standard WS-Notification WSDL.

   For more information, see Example: Subscribing a WS-Notification consumer, Implementing web services applications from existing WSDL files with JAX-WS and “Publishing the WSDL files for a WS-Notification application to a compressed file” on page 2984.

   Note: Instead of creating a new JAX-WS consumer application, you might choose to update the JAX-WS consumer application that you created in the subtask “Using JAX-WS clients and web services with migrated service points” on page 2979. The WSDL for a Version 7.0

WS-Notification service point contains a number of minor changes compared to a Version 6.1 service point, so you must modify your existing JAX-WS client application to take account of these changes. Specifically, you must regenerate the java proxy classes from the WSDL, and update any use of class names and methods that have changed. For example, there might be changes in the generated classes that include a port type or service from the WSDL.

b. Create a subscription for the new consumer service through the new Version 7.0 WS-Notification service point.
c. Prompt the WS-Notification service point to generate notifications (for example by using a WS-Notification client application).

The system transmits the notifications to the new JAX-WS consumer application correctly.

Sharing notifications between Version 6.1 and Version 7.0 WS-Notification service points:
About this task

You can configure WS-Notification so that notifications received through migrated Version 6.1 WS-Notification service points are published through the new Version 7.0 service. You might want to do this so that (for example) you can receive notifications through existing, unreliable connections then publish them through new connections made reliable through WS-ReliableMessaging. To enable this configuration, the new Version 7.0 WS-Notification service needs to use the same service integration bus topic space as the migrated Version 6.1 WS-Notification service. You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination. You configure a permanent topic namespace as a property of a WS-Notification service.

Procedure

1. Discover which bus topic spaces the migrated Version 6.1 WS-Notification service is using. If none, create a new permanent topic namespace to connect to a bus topic space. For more information, see “Modifying a Version 6.1 WS-Notification service” on page 2998.
2. Create a new permanent topic namespace for the new Version 7.0 WS-Notification service, that connects to the same bus topic space. For more information, see “Modifying a Version 7.0 WS-Notification service” on page 2991.

Results

Notifications received by either the new or migrated service point are now published to subscriptions made on either WS-Notification service.

Interacting at run time with WS-Notification

View (and in some cases delete) at run time the active items associated with WS-Notification service points.

Before you begin

This task assumes that you have a fully configured and operational WS-Notification service point.

About this task

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker,

subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

You can use the administrative console to interact at run time with the following active items associated with WS-Notification service points:

Publisher registrations
A runtime list of existing publisher registrations is provided. This includes, for each publisher registration, the information that was used to create it and when it will terminate.

Pull points
A runtime list is provided of the pull points that have been created. This includes, for each pull point, its current termination time and a link to the associated subscription.

Subscriptions
A runtime view is provided for subscriptions that have been created by applications. This includes, for each subscription, the information that was used to create it and an indication of its current state.

Administered subscribers
A runtime list is provided of the administered subscribers for a given WS-Notification service point. This includes, for each administered subscriber, an indication of its current state; for example whether the subscription was successfully initialized at start time.

For each WS-Notification service point, runtime information is available for subscriptions, registrations, pull points and administered subscribers. **For each WS-Notification service**, runtime information is available - aggregated for all service points for the service - for subscriptions, registrations and pull points. There is no aggregated view for administered subscribers at the WS-Notification service level.

To access and use the runtime information for active items associated with WS-Notification service points, use the administrative console to complete the following steps:

Procedure

1. Choose between information for a particular service point, or information aggregated for all service points for a particular service, by completing one of the following substeps:
   a. Optional: For runtime information for a particular service point, navigate to either **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations, pull points and administered subscribers for this WS-Notification service point.
   b. Optional: For runtime information aggregated for all service points for a particular service, navigate to either **Service integration -> WS-Notification -> Services -> *service_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name*** then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations and pull points for this WS-Notification service.
2. Click **Subscriptions** , **Publisher registrations**, **Pull points** or (for a particular service point) **Administered subscribers**.
   - If you click **Subscriptions**, a panel is displayed that lists the durable subscriptions that have been created by a WS-Notification service point in response to “Subscribe” requests from WS-Notification applications. You can view the subscription name (ID), topic and other information associated with the subscription, and you can view messages held on the durable subscription pending delivery. You can also delete subscriptions.

Note: Both the WS-Notification subscription and publisher registration resources include the concept of scheduled termination, in which the application indicates a period of time after which the resource is destroyed. “Badly-behaved” in this case describes an application that requests an infinite lifetime for a resource and then does not explicitly delete the resource before it goes away (never to return).

- If you click **Publisher registrations**, a panel is displayed that lists the publisher registrations that are currently in effect on this WS-Notification service or service point (that is, applications that have registered as publishers). You can view the basic properties of the registration record. You can also delete a publisher registration record.
- If you click **Pull points**, a panel is displayed that lists the pull points that are currently active on this WS-Notification service or service point. You can view basic properties of the pull point such as the subscription with which it is associated and the time at which it is currently set to expire, and you can navigate to the associated subscriptions where appropriate. You can also delete pull points.
- For a particular service point, if you click **Administered subscribers**, a panel is displayed that lists the administered subscribers that are currently in effect on this WS-Notification service point. You can use this information to see whether a given subscriber has been successfully initialized.

What to do next

For more detailed information about working with individual runtime panels, see the following topics:

- “Listing or deleting active WS-Notification subscriptions” on page 3027.
- “Listing or deleting active WS-Notification publisher registrations” on page 3029.
- “Listing or deleting active WS-Notification pull points” on page 3030.
- “Listing active WS-Notification administered subscribers” on page 3031.

Publishing the WSDL files for a WS-Notification application to a compressed file

Use the administrative console to download a compressed file with a `.zip` file extension that contains the published WSDL files for a WS-Notification application.

About this task

The ability to publish these WSDL files to a compressed file is particularly useful in the following circumstances:

- Writing a WS-Notification application that invokes web service operations against the NotificationBroker application, as described in Writing a WS-Notification application that does not expose a web service endpoint.
- Running the wsimport command against the exported `PublisherRegistrationManager.wsdl` file to generate a client stub for the PublisherRegistrationManager.
- Viewing the endpoint URLs to which WS-Notification applications connect, by looking in the WSDL file for the NotificationBroker application for Version 7.0 services, or the inbound service for Version 6.1 services.

Note:

When you run the wsimport command against the exported `PublisherRegistrationManager.wsdl` file you must include the `ibm-wsn-jaxws.xml` file as an argument to wsimport. If you omit this bindings file, the wsimport command fails with a naming conflict error concerning the ResourceNotDestroyedFault elements referred to in the `PublisherRegistrationManager.wsdl` file. For more information about why this exception occurs, see the following troubleshooting tip: The PublisherRegistrationManager.wsdl file is not successfully parsed by wsimport unless you include a JAX-WS bindings file.

The `ibm-wsn-jaxws.xml` file is located in the *app_server_root*`/util` directory. For example: `c:\was\util\ibm-wsn-jaxws.xml`. This bindings file expects to find the WSDL file to which it refers

in the same directory as itself, so before you run the wsimport command you must copy the bindings file to the directory that holds your `PublisherRegistrationManager.wsdl` file. Here is an example of how to run the wsimport command to include the `ibm-wsn-jaxws.xml` file:

```
c:\was\bin\wsimport -b ibm-wsn-jaxws.xml -keep PublisherRegistrationManager.wsdl
```

Procedure

1. Start the administrative console.
2. Navigate to the "Publish WSDL files to .zip file [Settings]" form for the WS-Notification application.

   For JAX-WS based Version 7.0 WS-Notification services, click one of the following paths:

   - **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> [Additional Properties] Publish WSDL files to zip**
   - **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> [Additional Properties] Publish WSDL files to zip**

   For JAX-RPC based Version 6.1 WS-Notification services, click one of the following paths:

   - **Service integration -> WS-Notification -> Services -> *service_name* -> [Related Items] Notification broker inbound service settings** > **[Additional Properties] Publish WSDL files to ZIP file**
   - **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Related Items] Notification broker inbound service settings** > **[Additional Properties] Publish WSDL files to ZIP file**
3. Click on the file name to download a zip file that contains the application's published WSDL files.

Configuring WS-Notification resources

You can configure individual WS-Notification resources by using the administrative console or by using the wsadmin tool.

About this task

To complete the tasks described in "Accomplishing common WS-Notification tasks" on page 2962, you configure the following types of WS-Notification resource:

- WS-Notification service
- WS-Notification service point
- WS-Notification administered subscriber
- Permanent WS-Notification topic namespace
- WS-Notification topic namespace document

You can configure individual WS-Notification resources by using the administrative console or by using the wsadmin tool, as described in the following tasks:

Procedure

- Configure WS-Notification resources by using the administrative console.
- Configure WS-Notification resources by using the wsadmin tool.
- Interact at run time with WS-Notification.
- Configure a JAX-WS client to resolve a WS-Notification service WSDL without following web links.

Configuring WS-Notification resources by using the administrative console

Use the administrative console to configure individual WS-Notification services, service points, administered subscribers, permanent topic namespaces, topic namespace documents and JAX-WS handlers.

Before you begin

Decide which method to use to configure these resources. You can configure WS-Notification resources by using the administrative console as described in this task, or by using the commands in the “WSNotificationCommands command group for the AdminTask object” on page 3053.

About this task

WS-Notification enables web services to use the publish and subscribe messaging pattern. For more information, see ../ae/cjwsn_overview.dita and “WS-Notification” on page 2964.

For high-level configuration of WS-Notification, see “Accomplishing common WS-Notification tasks” on page 2962. Use the steps for this task to configure individual instances of each type of WS-Notification resource.

You access the WS-Notification panels through the **Service integration** section of the administrative console navigation pane:

- To see a list of all the WS-Notification services, click **Service integration -> WS-Notification -> Services**.
- To see a list of all the WS-Notification services for a particular service integration bus, click **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**.

To support the Java API for XML-based Web Services (JAX-WS) and composition with WS-ReliableMessaging, you create your WS-Notification services as JAX-WS applications. The implementation of WS-Notification in WebSphere Application Server Version 6.1 uses JAX-RPC applications, so there are now two different implementations of the WS-Notification service and service point:

- **Version 7.0**: Use this type of service if you want to compose a JAX-WS WS-Notification service with web service qualities of service (QoS) via policy sets, or if you want to apply JAX-WS handlers to your WS-Notification service. This is the recommended type of service for new deployments. This WS-Notification option has been available in WebSphere Application Server from Version 7.0.
- **Version 6.1**: Use this type of service if you want to expose a JAX-RPC WS-Notification service that uses the same technology provided in WebSphere Application Server Version 6.1, including the ability to apply JAX-RPC handlers to the service. This WS-Notification option has been available in WebSphere Application Server from Version 6.1.

To configure specific WS-Notification resources, use the administrative console to complete any of the following tasks:

Procedure

- Configure a WS-Notification service.

  Complete any of the following tasks:
  - Create a new Version 7.0 WS-Notification service.
  - Modify a Version 7.0 WS-Notification service.
  - Create a new Version 6.1 WS-Notification service.
  - Modify a Version 6.1 WS-Notification service.
  - Delete WS-Notification services.
- Configure a WS-Notification service point.

  Complete any of the following tasks:
  - Create a new Version 7.0 WS-Notification service point.
  - Modify a Version 7.0 WS-Notification service point.
  - Create a new Version 6.1 WS-Notification service point.

  - Modify a Version 6.1 WS-Notification service point.
  - Delete WS-Notification service points.
- Configure a WS-Notification administered subscriber.
  Complete any of the following tasks:
  - Create a new WS-Notification administered subscriber.
  - Modify a WS-Notification administered subscriber.
  - Delete WS-Notification administered subscribers.
- Configure a permanent WS-Notification topic namespace.
  Complete any of the following tasks:
  - Create a new permanent WS-Notification topic namespace.
  - Show the properties of a permanent WS-Notification topic namespace.
  - Delete WS-Notification permanent topic namespaces.
- Configure a WS-Notification topic namespace document.
  Complete any of the following tasks:
  - Apply a WS-Notification topic namespace document.
  - Show the contents of a WS-Notification topic namespace document.
  - Delete WS-Notification topic namespace documents.
- Configure JAX-WS handlers and handler lists.
  Complete any of the following tasks:
  - Load JAX-WS handler classes.
  - Create a new JAX-WS handler configuration.
  - Modify an existing JAX-WS handler configuration.
  - Delete JAX-WS handler configurations.
  - Create a new JAX-WS handler list.
  - Modify an existing JAX-WS handler list.
  - Delete JAX-WS handler lists.

Creating a new Version 7.0 WS-Notification service:

Create a new WS-Notification service and the associated objects that form the infrastructure of the WS-Notification configuration. Use this type of service if you want to compose a JAX-WS WS-Notification service with web service qualities of service (QoS) via policy sets, or if you want to apply JAX-WS handlers to your WS-Notification service. This is the recommended type of service for new deployments. This WS-Notification option has been available in WebSphere Application Server from Version 7.0.

Before you begin

Decide which method to use to configure these resources. You can create a new Version 7.0 WS-Notification service by using the administrative console as described in this task, or by using the “createWSNService command” on page 3054.

This task assumes that you have an existing service integration bus, configured with at least one bus member.

You usually configure one WS-Notification service for a service integration bus, but you can configure more than one. For more information, see Reasons to create multiple WS-Notification services in a bus.

Defining a Version 7.0 WS-Notification service is not the same as exposing a NotificationBroker (WSDL) port to which web services applications can connect. To do this, create one or more Version 7.0 WS-Notification service points as described in this task.

About this task

A WS-Notification service provides the ability to expose some or all of the messaging resources defined on a service integration bus for use by WS-Notification applications.

To support the Java API for XML-based Web Services (JAX-WS) and composition with WS-ReliableMessaging, you create your WS-Notification services as JAX-WS applications, then use this task to create a Version 7.0 WS-Notification service, one or more service points, and (optionally) a permanent topic namespace.

You can also apply JAX-WS handler lists to WS-Notification service points (for inbound invocation handling) and WS-Notification services (for outbound invocation handling).

When you create a Version 7.0 WS-Notification service, the wizard creates and deploys a JAX-WS based provider application. This application exposes the WS-Notification web service interfaces for each of the three WS-Notification service roles:

- Notification broker
- Subscription manager
- Publisher registration manager

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The "WS-Notification services [Collection]" on page 3047 form is displayed.
3. In the content pane, click **New**. The "New WS-Notification service" wizard is displayed. For more information about the properties that you set with the wizard, see "WS-Notification services [Settings]" on page 3047.
4. Step 1: Configure name, description, service integration bus and dynamic topic namespace settings.
   a. Enter your chosen name and an optional description.

      The name forms part of the endpoint on which the service is exposed (that is, the URL used to access the WS-Notification service points that are defined under the service). For Version 6.1 WS-Notification services, the service name is unique within a bus. For Version 7.0 WS-Notification services the service name is unique within the cell, which matches the administration model used for policy sets and therefore supports composition of Version 7.0 WS-Notification services with WS-ReliableMessaging.

   b. Select or deselect the option **Enable dynamic topic namespaces?**.

      Indicates whether dynamic topic namespaces can be used within the WS-Notification service. That is, whether this service allows dynamic topic namespaces to be created at run time. For more information, see Dynamic topic namespace.

      Use this option to tightly control the topic namespaces that are used when connecting to a particular WS-Notification service (for example for security or auditing requirements). If you deselect this option, any applications that connect to the WS-Notification service and request topics from a dynamic topic namespace are stopped from publishing or receiving messages.

      All messages published to a dynamic topic namespace are inserted with the default message reliability setting of reliable persistent. If this value is not acceptable, create a permanent topic namespace and manually configure the attribute to the appropriate value.

      Note: The dynamic topic namespaces used on a particular WS-Notification service are backed by a service integration bus topic space that is created automatically when you create the topic namespace. The syntax of topics used within this topic space is internal to the WS-Notification service implementation.

c. Select or deselect the option **Requires registration**.

   Indicates whether publisher applications are required to register with the broker before they can publish notifications.

d. Select a service integration bus from the drop-down list.

e. Click **Next**.

5. Step 2: Select WS-Notification service type.

   Select **Version 7.0** as the type of service that you want to create.

6. Step 3: Configure handler and web service policy settings.

   These settings are applied to the event notifications exchanged with WS-Notification client applications.

   a. Optional: Choose a JAX-WS handler list.

      The JAX-WS handler list that is applied to outbound requests from the WS-Notification service. A handler list defines the handlers that are applied when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume). For more information about handler lists, see “Configuring JAX-WS handlers” on page 2972.

   b. Enable or clear the **Query WSDL** option.

      Indicates whether the Version 7.0 WS-Notification service queries the WSDL of other WS-Notification web services when interacting with them. By default, this option is enabled. By clearing this option, you can improve performance by avoiding expensive WSDL queries. However, you should note the following considerations when WSDL querying is not enabled:

      - WS-Notification attempts to discover binding information (which is usually discovered through the WSDL) by using other means. WS-Notification uses the SOAP version associated with the WS-Notification service point where subscriptions were made (by other web services), or where administered subscriptions were created (by an administrator).
      - There are some circumstances in which WS-Notification might be unable to determine binding information. This can happen when cleaning up subscriptions where the associated service point has been deleted and configuration information is no longer available. Under these circumstances WS-Notification makes a “best guess” at binding information to use to clean up the subscriptions.
      - There is one scenario where incorrect binding information is used. That is, when a subscriber subscribes to use a particular SOAP binding, on behalf of a NotificationConsumer that expects notifications through a different SOAP binding.

   c. Enter a dynamic topic space name.

      The name of the service integration bus topic space to be used as the dynamic topic space for this WS-Notification service. That is, the name of the bus topic space that is used to host the ad-hoc topic namespace, and to host dynamic topic namespaces if they are permitted. A default name of `WSN_dynamic_this_service_name` is offered.

   d. Click **Next**.

7. Step 4: Create WS-Notification service points.

   A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service. For more information, see WS-Notification service point.

   a. Select **Yes** to create a new WS-Notification service point, then click **Next**.

      A WS-Notification service must have at least one service point.

b. Supply a name and (optional) description for the WS-Notification service point, and from the drop-down list select the bus member on which the service point is to be configured, then click **Next**.

The service point name forms part of the URL used to access the service point. On a single server system there is only one bus member in the list.

c. Select the transport settings for the new service point.

Service point accessed via HTTP proxy
If the service point is accessed through a proxy, select the check box, and type the root of the externally visible endpoint address URL for web services accessed through this endpoint.

The URL for the proxy is used to populate the WSDL endpoint address fields when publishing WSDL files to a compressed file.

SOAP Version
Select the version of SOAP that is supported by the service point. This affects the WSDL definition that is exposed by the web service.

d. Optional: Select the JAX-WS handler list settings for the new service point.

NotificationBroker JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the NotificationBroker endpoint of the WS-Notification service point.

SubscriptionManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the SubscriptionManager endpoint of the WS-Notification service point.

PublisherRegistrationManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the PublisherRegistrationManager endpoint of the WS-Notification service point.

e. Click **Next**. The new service point is added to the list of service points for this WS-Notification service.

f. Optional: To create another service point, repeat the previous substeps.

g. When you have finished creating service points for this WS-Notification service, select **No** for the option to create another service point, then click **Next**.

8. Optional: Step 5: Create permanent topic namespaces.

For more information, see Permanent topic namespace. When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents.

a. Select **Yes** to create a new permanent topic namespace, then click **Next**.

b. Enter a name for the permanent topic namespace.

This is the URI by which WS-Notification applications refer to topics hosted by this namespace.

c. Associate this new permanent topic namespace with the service integration bus topic space that you want to use to publish and receive messages.

From the service integration bus topic space drop-down list, complete one of the following actions:

- Select the name of an existing bus topic space.
- Select the option to `Create a new topic space`, then enter a name for the new topic space.

d. Select from the drop-down list the service integration bus reliability (quality of service) that is assigned to messages published through this topic namespace.

You can choose one of five values, each representing one of the service integration bus message reliability levels. The default value is `reliable persistent`, which is the value used by default for JMS Persistent messages.

e. Click **Next**.

The new permanent topic namespace is added to a list of permanent topic namespaces for this Version 7.0 WS-Notification service, and you are asked whether you want to create another permanent topic namespace (default is **Yes**).

f. Optional: To create another permanent topic namespace, repeat the previous substeps.

g. When you have finished creating permanent topic namespaces for this Version 7.0 WS-Notification service, select **No** for the option to create another permanent topic namespace, then click **Next**.

9. Step 6: Summary.

Check that the summary of the actions taken by the wizard is as you expected, then click **Finish**. If the processing completes successfully, the list of Version 7.0 WS-Notification services is updated to include the new Version 7.0 WS-Notification service. Otherwise, an error message is displayed.

10. Save your changes to the master configuration.

11. Optional: Restart the server if either of the following conditions apply:
    - A new bus or new bus member has been created as part of this task.
    - **Configuration reload** is not enabled for the bus.

What to do next

To undertake advanced configuration tasks for this WS-Notification service (for example, adding additional service points and applying topic namespace documents to permanent topic namespaces), see “Modifying a Version 7.0 WS-Notification service.”

To undertake advanced configuration tasks for the WS-Notification service point that you created as part of this task (for example, adding administered subscribers, publishing WSDL files to a compressed file, and configuring the enterprise application associated with this service point), see “Modifying a Version 7.0 WS-Notification service point” on page 3003.

To configure this WS-Notification service or service point with web service qualities of service (QoS) such as reliability or security, see “Configuring a Version 7.0 WS-Notification service with Web service QoS” on page 2974.

Modifying a Version 7.0 WS-Notification service:

Modify the **description**, **Enabled dynamic topic namespaces?**, **Requires registration**, **JAX-WS handler list** and **Query WSDL** properties of a Version 7.0 WS-Notification service, and follow links to complete advanced configuration such as adding additional WS-Notification service points, applying topic namespace documents to permanent topic namespaces, and applying policy sets to enable WS-ReliableMessaging.

About this task

A WS-Notification service provides the ability to expose some or all of the messaging resources defined on a service integration bus for use by WS-Notification applications.

A handler list defines the handlers that are applied when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume).

When you create a Version 7.0 WS-Notification service, the wizard creates and deploys a JAX-WS based provider application. This application exposes the WS-Notification web service interfaces for each of the three WS-Notification service roles:

- Notification broker
- Subscription manager
- Publisher registration manager

You can also configure custom properties to specify a timeout time for outbound requests, and to determine the strictness of the syntax checking of topics used under this Version 7.0 WS-Notification service.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The "WS-Notification services [Collection]" on page 3047 form is displayed.
3. In the content pane, click the name of a Version 7.0 WS-Notification service in the list. The current settings for this Version 7.0 WS-Notification service are displayed in the "WS-Notification services [Settings]" on page 3047 panel.
4. Modify the following general properties:

   Description
   An optional description of the WS-Notification service.

   Enable dynamic topic namespaces?

   Indicates whether dynamic topic namespaces can be used within the WS-Notification service. That is, whether this service allows dynamic topic namespaces to be created at run time. For more information, see Dynamic topic namespace.

   Use this option to tightly control the topic namespaces that are used when connecting to a particular WS-Notification service (for example for security or auditing requirements). If you deselect this option, any applications that connect to the WS-Notification service and request topics from a dynamic topic namespace are stopped from publishing or receiving messages.

   All messages published to a dynamic topic namespace are inserted with the default message reliability setting of reliable persistent. If this value is not acceptable, create a permanent topic namespace and manually configure the attribute to the appropriate value.

   Note: The dynamic topic namespaces used on a particular WS-Notification service are backed by a service integration bus topic space that is created automatically when you create the topic namespace. The syntax of topics used within this topic space is internal to the WS-Notification service implementation.

   Requires registration
   Indicates whether publisher applications are required to register with the broker before they can publish notifications.

   JAX-WS handler list

   The JAX-WS handler list that is applied to outbound requests from the WS-Notification service.

   A handler list defines the handlers that are applied when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume). For more information about handler lists, see "Configuring JAX-WS handlers" on page 2972.

   Query WSDL

Indicates whether the Version 7.0 WS-Notification service queries the WSDL of other WS-Notification web services when interacting with them. By default, this option is enabled. By clearing this option, you can improve performance by avoiding expensive WSDL queries. However, you should note the following considerations when WSDL querying is not enabled:

- WS-Notification attempts to discover binding information (which is usually discovered through the WSDL) by using other means. WS-Notification uses the SOAP version associated with the WS-Notification service point where subscriptions were made (by other web services), or where administered subscriptions were created (by an administrator).
- There are some circumstances in which WS-Notification might be unable to determine binding information. This can happen when cleaning up subscriptions where the associated service point has been deleted and configuration information is no longer available. Under these circumstances WS-Notification makes a “best guess” at binding information to use to clean up the subscriptions.
- There is one scenario where incorrect binding information is used. That is, when a subscriber subscribes to use a particular SOAP binding, on behalf of a NotificationConsumer that expects notifications through a different SOAP binding.

5. Modify the additional properties:

WS-Notification service points

Select this link to configure the deployment of WS-Notification service points on one or more servers. For more information, see “Creating a new Version 7.0 WS-Notification service point” on page 3001 or “Modifying a Version 7.0 WS-Notification service point” on page 3003.

Permanent topic namespaces

Select this link to configure permanent topic namespaces for the WS-Notification service. For more information, see Permanent topic namespace. When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents.

Custom properties

Select this link to configure additional custom properties for this WS-Notification service. These custom properties are name and value pairs that you can use to set internal system configuration properties. In each pair, the name is a property key and the value is a string value.

To specify a timeout time for outbound requests sent from this WS-Notification service, set the following custom property:

outbound.timeout

The value of this property is the timeout time in milliseconds. If the property is not set, a default timeout of 2 minutes is used.

To determine the strictness of the syntax checking of topics used under this WS-Notification service, set the following custom property:

com.ibm.ws.sib.wsn.strictTopicChecking

Valid values for this property are TRUE and FALSE:

- If the property value is set to TRUE, the topic syntax rules defined in the WS-Topics standard are strictly enforced. Note that there is a performance cost compared to the default setting, because each character of a topic is validated against a large list of permitted Unicode characters.
- If the property is omitted or set to FALSE, syntax checking only ensures that the basic topic structure is valid, and character checking is relaxed to allow any character except * (asterisk) and . (dot) as a topic name part.

Outbound request policy sets and bindings

The outbound request policy sets and bindings for the two WS-Notification service clients associated with this WS-Notification service. For reliable web service transmission of notification messages, use this option to associate the WS-Notification service client with a policy set that enables WS-ReliableMessaging.

For more information, see “Configuring a Version 7.0 WS-Notification service with Web service QoS” on page 2974.

6. Apply any changes, then click **OK**. If the processing completes successfully, the list of WS-Notification services is redisplayed. Otherwise, an error message is displayed.
7. Save your changes to the master configuration. You need not restart the server for the changes to fully take effect if **configuration reload** is enabled for the service integration bus.

Deleting WS-Notification services:

Remove all configuration associated with one or more WS-Notification services. A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

Before you begin

Decide which method to use to configure these resources. You can delete a WS-Notification service by using the administrative console as described in this task, or by using the “deleteWSNService command” on page 3057.

About this task

When you delete a Version 6.1 WS-Notification service, the associated inbound services and inbound ports are also deleted. The associated endpoint listeners are deleted if they were created in the process of creating a WS-Notification service point and are not used by any other configuration object. When you delete a Version 7.0 WS-Notification service, the associated service point provider applications and service points are also deleted. Do not delete these resources manually, because this might leave the associated configuration information in an inconsistent state.

For both types of WS-Notification service, you choose whether the associated service integration bus topic spaces are deleted.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. A list of all the WS-Notification services is displayed in a “WS-Notification services [Collection]” on page 3047 form.
3. Select the check box for every WS-Notification service that you want to remove.
4. Click **Delete**. A panel is displayed asking if the service integration bus topic spaces associated with the WS-Notification service (including those associated with all the WS-Notification topic namespaces) should also be deleted.
5. Optional: Select the check box if you want to delete the associated service integration bus topic spaces.

   Note: Deleting a service integration bus topic space causes exception messages to be generated for WS-Notification applications that reference the topic space, as described in Failures as a result of changes in topic space and topic namespace configurations.
6. Click **Delete**. If the processing completes successfully, the list of WS-Notification services is updated. Otherwise, an error message is displayed.
7. Save your changes to the master configuration.

Creating a new Version 6.1 WS-Notification service:

Create a new WS-Notification service and the associated objects that form the infrastructure of the WS-Notification configuration. Use this type of service if you want to expose a JAX-RPC WS-Notification service that uses the same technology provided in WebSphere Application Server Version 6.1, including the ability to apply JAX-RPC handlers to the service. This WS-Notification option has been available in WebSphere Application Server from Version 6.1.

Before you begin

Ensure that you have successfully configured an SDO repository, as described in “Installing and configuring the SDO repository” on page 2772. The SDO repository is used to store WSDL documents during the creation of the WS-Notification service. If you do not configure the repository, an error message appears when you create the service.

Decide which method to use to configure these resources. You can create a new Version 6.1 WS-Notification service by using the administrative console as described in this task, or by using the “createWSNService command” on page 3054.

This task assumes that you have an existing service integration bus, configured with at least one bus member.

You usually configure one WS-Notification service for a service integration bus, but you can configure more than one. For more information, see Reasons to create multiple WS-Notification services in a bus.

Defining a WS-Notification service on a bus is not the same as exposing a NotificationBroker (WSDL) port to which web services applications can connect. To do this, create one or more WS-Notification service points as described in this task.

About this task

A WS-Notification service provides the ability to expose some or all of the messaging resources defined on a service integration bus for use by WS-Notification applications.

A JAX-RPC handler list and WS-Security bindings define the parameters and security policy that are used when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume).

When you create a Version 6.1 WS-Notification service, the wizard configures three service integration bus inbound services for the WS-Notification service, one for each of the three WS-Notification service roles:

- Notification broker
- Subscription manager
- Publisher registration manager

These inbound services are defined on the same service integration bus as the Version 6.1 WS-Notification service, and each of these inbound services refers to the same bus destination.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The “WS-Notification services [Collection]” on page 3047 form is displayed.

3. In the content pane, click **New**. The “New WS-Notification service” wizard is displayed. For more information about the properties that you set with the wizard, see “WS-Notification services [Settings]” on page 3047.
4. Step 1: Configure name, description, service integration bus and dynamic topic namespace settings.
   a. Enter your chosen name and an optional description.

      The name forms part of the endpoint on which the service is exposed (that is, the URL used to access the WS-Notification service points that are defined under the service). For Version 6.1 WS-Notification services, the service name is unique within a bus. For Version 7.0 WS-Notification services the service name is unique within the cell, which matches the administration model used for policy sets and therefore supports composition of Version 7.0 WS-Notification services with WS-ReliableMessaging.
   b. Select or deselect the option **Enable dynamic topic namespaces?**.

      Indicates whether dynamic topic namespaces can be used within the WS-Notification service. That is, whether this service allows dynamic topic namespaces to be created at run time. For more information, see Dynamic topic namespace.

      Use this option to tightly control the topic namespaces that are used when connecting to a particular WS-Notification service (for example for security or auditing requirements). If you deselect this option, any applications that connect to the WS-Notification service and request topics from a dynamic topic namespace are stopped from publishing or receiving messages.

      All messages published to a dynamic topic namespace are inserted with the default message reliability setting of reliable persistent. If this value is not acceptable, create a permanent topic namespace and manually configure the attribute to the appropriate value.

      Note: The dynamic topic namespaces used on a particular WS-Notification service are backed by a service integration bus topic space that is created automatically when you create the topic namespace. The syntax of topics used within this topic space is internal to the WS-Notification service implementation.
   c. Select or deselect the option **Requires registration**.

      Indicates whether publisher applications are required to register with the broker before they can publish notifications.
   d. Select a service integration bus from the drop-down list.
   e. Click **Next**.
5. Step 2: Select WS-Notification service type.

   Select **Version 6.1** as the type of service that you want to create.
6. Step 3: Configure handler and web service policy settings.

   These settings are applied to the event notifications exchanged with WS-Notification client applications.
   a. Optional: Choose a JAX-RPC handler list.

      The JAX-RPC handler list that is applied to outbound requests from the WS-Notification service - for example the broker delivering notifications to a consumer. For more information about handler lists, see “Working with JAX-RPC handlers and clients” on page 2799.
   b. Optional: Choose a WS-Security configuration and bindings:

      Outbound security request binding
      The security binding to be used with consumer notifications and remote publisher requests sent by this WS-Notification service.

      Outbound security response binding
      The security binding to be used with remote publisher responses received by this WS-Notification service.

      Outbound security configuration
      Specifies the details of how security is applied to requests and responses.

For more information about Web Services Security resources, see Configuring secure transmission of SOAP messages by using WS-Security.

c. Enter a dynamic topic space name.

The name of the service integration bus topic space to be used as the dynamic topic space for this WS-Notification service. That is, the name of the bus topic space that is used to host the ad-hoc topic namespace, and to host dynamic topic namespaces if they are permitted. A default name of `WSN_dynamic_this_service_name` is offered.

d. Click **Next**.

7. Step 4: Create WS-Notification service points.

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service. For more information, see WS-Notification service point.

a. Select **Yes** to create a new WS-Notification service point, then click **Next**.

A WS-Notification service must have at least one service point.

b. Supply a name and (optional) description for the WS-Notification service point, and from the drop-down list select the bus member on which the service point is to be configured, then click **Next**.

The service point name forms part of the URL used to access the service point (that is, the address of the web service that is exposed on the chosen server). On a single server system there is only one bus member in the list.

c. Select a listener application to use to expose the service. Either select an existing endpoint listener for this bus member, or **Create a new endpoint listener**.

For more information, see “Creating a new endpoint listener configuration” on page 2789.

d. Click **Next**. The new service point is added to the list of service points for this WS-Notification service.

e. Optional: To create another service point, repeat the previous substeps.

f. When you have finished creating service points for this WS-Notification service, select **No** for the option to create another service point, then click **Next**.

8. Optional: Step 5: Create permanent topic namespaces.

When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents. For more information, see Permanent topic namespace.

a. Select **Yes** to create a new permanent topic namespace, then click **Next**.

b. Enter a name for the permanent topic namespace.

This is the URI by which WS-Notification applications refer to topics hosted by this namespace.

c. Associate this new permanent topic namespace with the service integration bus topic space that you want to use to publish and receive messages.

From the service integration bus topic space drop-down list, complete one of the following actions:

- Choose the name of an existing bus topic space.
- Choose the option to `Create a new topic space`, then enter a name for the new topic space.

d. Select from the drop-down list the service integration bus reliability (quality of service) that is assigned to messages published through this topic namespace.

You can choose one of five values, each representing one of the service integration bus message reliability levels. The default value is `reliable persistent`, which is the value used by default for JMS Persistent messages.

e. Click **Next**.

The new permanent topic namespace is added to a list of permanent topic namespaces for this WS-Notification service, and you are asked whether you want to create another permanent topic namespace (default is **Yes**).

f. Optional: To create another permanent topic namespace, repeat the previous substeps.

g. When you have finished creating permanent topic namespaces for this WS-Notification service, select **No** for the option to create another permanent topic namespace, then click **Next**.

9. Step 6: Summary.

Check that the summary of the actions taken by the wizard is as you expected, then click **Finish**. If the processing completes successfully, the list of WS-Notification services is updated to include the new Version 6.1 WS-Notification service. Otherwise, an error message is displayed.

10. Save your changes to the master configuration.

11. Optional: Restart the server if either of the following conditions apply:
    - A new bus or new bus member has been created as part of this task.
    - **Configuration reload** is not enabled for the bus.

What to do next

To undertake advanced configuration tasks for this WS-Notification service (for example adding additional WS-Notification service points, or applying topic namespace documents to permanent topic namespaces), see “Modifying a Version 6.1 WS-Notification service.”

Modifying a Version 6.1 WS-Notification service:

Modify the **description**, **Enabled dynamic topic namespaces?** and **Requires registration** properties of a WS-Notification service, and follow links to complete advanced configuration of the WS-Notification service such as adding additional WS-Notification service points, or applying topic namespace documents to permanent topic namespaces.

About this task

A WS-Notification service provides the ability to expose some or all of the messaging resources defined on a service integration bus for use by WS-Notification applications.

A JAX-RPC handler list and WS-Security bindings define the parameters and security policy that are used when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume).

When you create a Version 6.1 WS-Notification service, the wizard configures three service integration bus inbound services for the WS-Notification service, one for each of the three WS-Notification service roles:

- Notification broker
- Subscription manager
- Publisher registration manager

These inbound services are defined on the same service integration bus as the Version 6.1 WS-Notification service, and each of these inbound services refers to the same bus destination.

You can make web services-specific modifications to the WS-Notification service behavior by modifying the three associated inbound services. You can also configure a custom property that determines the strictness of the syntax checking of topics used under this WS-Notification service.

To modify a WS-Notification service use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. The "WS-Notification services [Collection]" on page 3047 form is displayed.
2. In the content pane, click the name of a WS-Notification service in the list. The current settings for this WS-Notification service are displayed in the "WS-Notification services [Settings]" on page 3047 panel.
3. Modify the following general properties:

   Description
   An optional description of the WS-Notification service.

   Enable dynamic topic namespaces?

   Indicates whether dynamic topic namespaces can be used within the WS-Notification service. That is, whether this service allows dynamic topic namespaces to be created at run time. For more information, see Dynamic topic namespace.

   Use this option to tightly control the topic namespaces that are used when connecting to a particular WS-Notification service (for example for security or auditing requirements). If you deselect this option, any applications that connect to the WS-Notification service and request topics from a dynamic topic namespace are stopped from publishing or receiving messages.

   All messages published to a dynamic topic namespace are inserted with the default message reliability setting of reliable persistent. If this value is not acceptable, create a permanent topic namespace and manually configure the attribute to the appropriate value.

   Note: The dynamic topic namespaces used on a particular WS-Notification service are backed by a service integration bus topic space that is created automatically when you create the topic namespace. The syntax of topics used within this topic space is internal to the WS-Notification service implementation.

   Requires registration
   Indicates whether publisher applications are required to register with the broker before they can publish notifications.
4. Modify the JAX-RPC handler list and Web Services Security settings. These settings are applied to the event notifications exchanged with WS-Notification client applications. For more information about handler lists, see "Working with JAX-RPC handlers and clients" on page 2799. For more information about Web Services Security resources, see Configuring secure transmission of SOAP messages by using WS-Security.

   JAX-RPC handler list
   The JAX-RPC handler list that is applied to outbound requests from the WS-Notification service - for example the broker delivering notifications to a consumer.

   Outbound security request binding
   The security binding to be used with consumer notifications and remote publisher requests sent by this WS-Notification service.

   Outbound security response binding
   The security binding to be used with remote publisher responses received by this WS-Notification service.

   Outbound security configuration
   Specifies the details of how security is applied to requests and responses.

5. Modify the additional properties:

   WS-Notification service points
   Select this link to configure the deployment of WS-Notification service points on one or more servers. For more information, see “Modifying a Version 6.1 WS-Notification service point” on page 3006.

   Permanent topic namespaces
   Select this link to configure permanent topic namespaces for the WS-Notification service. For more information, see Permanent topic namespace. When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents.

   Custom properties
   Select this link to configure additional custom properties for this WS-Notification service. These custom properties are name and value pairs that you can use to set internal system configuration properties. In each pair, the name is a property key and the value is a string value.

   To specify a timeout time for outbound requests sent from this WS-Notification service, set the following custom property:

   `outbound.timeout`

   The value of this property is the timeout time in milliseconds. If the property is not set, a default timeout of 2 minutes is used.

   To determine the strictness of the syntax checking of topics used under this WS-Notification service, set the following custom property:

   `com.ibm.ws.sib.wsn.strictTopicChecking`

   Valid values for this property are TRUE and FALSE:

   - If the property value is set to TRUE, the topic syntax rules defined in the WS-Topics standard are strictly enforced. Note that there is a performance cost compared to the default setting, because each character of a topic is validated against a large list of permitted Unicode characters.
   - If the property is omitted or set to FALSE, syntax checking only ensures that the basic topic structure is valid, and character checking is relaxed to allow any character except * (asterisk) and . (dot) as a topic name part.

6. Modify the inbound service settings for the notification broker, subscription manager or publisher registration manager. For more information, see “Modifying an existing inbound service configuration” on page 2780.
7. Apply any changes, then click **OK**. If the processing completes successfully, the list of WS-Notification services is redisplayed. Otherwise, an error message is displayed.
8. Save your changes to the master configuration. You need not restart the server for the changes to fully take effect if **configuration reload** is enabled for the service integration bus.

Deleting WS-Notification services:

Remove all configuration associated with one or more WS-Notification services. A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

Before you begin

Decide which method to use to configure these resources. You can delete a WS-Notification service by using the administrative console as described in this task, or by using the “deleteWSNService command” on page 3057.

About this task

When you delete a Version 6.1 WS-Notification service, the associated inbound services and inbound ports are also deleted. The associated endpoint listeners are deleted if they were created in the process of creating a WS-Notification service point and are not used by any other configuration object. When you delete a Version 7.0 WS-Notification service, the associated service point provider applications and service points are also deleted. Do not delete these resources manually, because this might leave the associated configuration information in an inconsistent state.

For both types of WS-Notification service, you choose whether the associated service integration bus topic spaces are deleted.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> WS-Notification -> Services** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**. A list of all the WS-Notification services is displayed in a “WS-Notification services [Collection]” on page 3047 form.
3. Select the check box for every WS-Notification service that you want to remove.
4. Click **Delete**. A panel is displayed asking if the service integration bus topic spaces associated with the WS-Notification service (including those associated with all the WS-Notification topic namespaces) should also be deleted.
5. Optional: Select the check box if you want to delete the associated service integration bus topic spaces.

   Note: Deleting a service integration bus topic space causes exception messages to be generated for WS-Notification applications that reference the topic space, as described in Failures as a result of changes in topic space and topic namespace configurations.
6. Click **Delete**. If the processing completes successfully, the list of WS-Notification services is updated. Otherwise, an error message is displayed.
7. Save your changes to the master configuration.

Creating a new Version 7.0 WS-Notification service point:

You can add an additional Version 7.0 WS-Notification service point to an existing Version 7.0 WS-Notification service. A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

Before you begin

Decide which method to use to configure these resources. You can create a new Version 7.0 WS-Notification service point by using the administrative console as described in this task, or by using the “createWSNServicePoint command” on page 3061.

About this task

You can define any number of WS-Notification service points for a given WS-Notification service. Each service point defined for the same WS-Notification service represents an alternative entry point to the service. Event notifications published to a particular WS-Notification service point are received by all applications connected to any service point of the same WS-Notification service (subject to subscription on the correct topic) regardless of the particular service point to which they are connected. For more information, see Reasons to create multiple WS-Notification service points.

When you create a Version 7.0 WS-Notification service point you select a bus member on which the WS-Notification service point is configured. You also choose the SOAP version that is supported by the service point, and (optionally) apply JAX-WS handler lists to the NotificationBroker, SubscriptionManager or PublisherRegistrationManager that are exposed through this service point.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**. The "WS-Notification service points [Collection]" on page 3043 form is displayed. This form shows all the Version 7.0 WS-Notification service points configured for this Version 7.0 WS-Notification service.
3. In the content pane, click **New**. The New WS-Notification service point wizard is displayed. For more information about the properties that you set with the wizard, see "WS-Notification service points [Settings]" on page 3043.
4. Step 1: Configure name, description and select a bus member.
   a. Supply a name and (optional) description for the WS-Notification service point. The service point name forms part of the URL used to access the service point.
   b. From the drop-down list, select the bus member on which the service point is to be configured. On a single server system there is only one bus member in the list.
   c. Click **Next**.
5. Step 2: Define transport settings.
   a. Select the transport settings for the new service point.

      Service point accessed via HTTP proxy
      If the service point is accessed through a proxy, select the check box, and type the root of the externally visible endpoint address URL for web services accessed through this endpoint.

      The URL for the proxy is used to populate the WSDL endpoint address fields when publishing WSDL files to a compressed file.

      SOAP Version
      Select the version of SOAP that is supported by the service point. This affects the WSDL definition that is exposed by the web service.

   b. Optional: Select the JAX-WS handler list settings for the new service point.

      NotificationBroker JAX-WS handler list

      The JAX-WS handler list that is applied to inbound requests from an application to the NotificationBroker endpoint of the WS-Notification service point.

      SubscriptionManager JAX-WS handler list

      The JAX-WS handler list that is applied to inbound requests from an application to the SubscriptionManager endpoint of the WS-Notification service point.

PublisherRegistrationManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the PublisherRegistrationManager endpoint of the WS-Notification service point.

6. Click **Finish**. If the processing completes successfully, the list of Version 7.0 WS-Notification service points for this Version 7.0 WS-Notification service is updated to include the new service point. Otherwise, an error message is displayed.
7. Save your changes to the master configuration.

What to do next

To undertake advanced configuration tasks for this WS-Notification service point (for example, adding administered subscribers, publishing WSDL files to a compressed file, and configuring the enterprise application associated with this service point), see "Modifying a Version 7.0 WS-Notification service point."

To configure this WS-Notification service point with web service qualities of service (QoS) such as reliability or security, see "Configuring a Version 7.0 WS-Notification service with Web service QoS" on page 2974.

You can also use the administrative console to work with runtime information for service points. For more information, see "Interacting at run time with WS-Notification" on page 2982.

Modifying a Version 7.0 WS-Notification service point:

Modify the **description**, **SOAP Version**, and **JAX-WS handler list** properties of a Version 7.0 WS-Notification service point, and follow links to complete advanced configuration such as modifying the administered subscribers, applying policy sets to enable WS-ReliableMessaging, publishing the WSDL files to compressed files, and configuring the enterprise application that is associated with this service point.

About this task

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**. The "WS-Notification service points [Collection]" on page 3043 form is displayed. This form shows all the service points configured for this Version 7.0 WS-Notification service.
3. In the content pane, click the name of a Version 7.0 WS-Notification service point in the list. The current settings for this Version 7.0 WS-Notification service point are displayed in the "WS-Notification service points [Settings]" on page 3043 form.
4. Modify the following general properties:

   Description

   An optional description of the WS-Notification service point.

Associated bus member
The name of the bus member on which this WS-Notification service point is deployed.

SOAP Version

Defines the version of SOAP supported by the service point. This affects the WSDL definition that will be exposed by the web service. Permitted values are 1.1 for SOAP 1.1 (the default), and 1.2 for SOAP 1.2.

NotificationBroker JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the NotificationBroker endpoint of the WS-Notification service point.

SubscriptionManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the SubscriptionManager endpoint of the WS-Notification service point.

PublisherRegistrationManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the PublisherRegistrationManager endpoint of the WS-Notification service point.

5. Modify the additional properties:

Administered subscribers
An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time. For more information, see "Modifying a WS-Notification administered subscriber" on page 3010.

Custom properties
Select this link to configure additional custom properties for this WS-Notification service point.

Policy set configuration
The policy set configuration associated with this WS-Notification service point. You can configure policy set and binding information for each port relating to this service point. For more information, see "Configuring a Version 7.0 WS-Notification service with Web service QoS" on page 2974.

Publish WSDL files to zip
Publish the WSDL files for this service point to a compressed file.

Note: When you run the wsimport command against the exported `PublisherRegistrationManager.wsdl` file you must include the `ibm-wsn-jaxws.xml` file as an argument to wsimport.

For more information, see "Publishing the WSDL files for a WS-Notification application to a compressed file" on page 2984.

Service point application
The application associated with this WS-Notification service point. For Version 7.0 WS-Notification services, enterprise applications are used to expose the web services associated with the WS-Notification service.

6. Apply any changes, then click **OK**. If the processing completes successfully, the list of WS-Notification service points for this Version 7.0 WS-Notification service is redisplayed. Otherwise, an error message is displayed.
7. Save your changes to the master configuration.

What to do next

You can also use the administrative console to work with runtime information for service points. For more information, see "Interacting at run time with WS-Notification" on page 2982.

Deleting WS-Notification service points:

Delete one or more WS-Notification service points and the associated resources (Version 7.0 service point provider applications, or Version 6.1 inbound services and endpoint listeners).

Before you begin

Decide which method to use to configure these resources. You can delete a WS-Notification service point by using the administrative console as described in this task, or by using the “deleteWSNServicePoint command” on page 3064.

Do not use this task to delete service points as part of the process of deleting a WS-Notification service. When you delete a WS-Notification service, the associated service points and resources are automatically deleted.

About this task

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. When you delete a WS-Notification service point, the associated resources (Version 7.0 service point provider applications, or Version 6.1 inbound services and endpoint listeners) are automatically deleted. Do not delete these resources manually, because this might leave the associated configuration information in an inconsistent state.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**. A list of all the WS-Notification service points is displayed in a “WS-Notification service points [Collection]” on page 3043 form.
3. Select the check box for every WS-Notification service point that you want to remove.
4. Click **Delete**. If the processing completes successfully, the list of WS-Notification service points for this WS-Notification service is updated. Otherwise, an error message is displayed.
5. Save your changes to the master configuration.

Creating a new Version 6.1 WS-Notification service point:

You can add an additional Version 6.1 WS-Notification service point to an existing Version 6.1 WS-Notification service. A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

Before you begin

Decide which method to use to configure these resources. You can create a new Version 6.1 WS-Notification service point by using the administrative console as described in this task, or by using the “createWSNServicePoint command” on page 3061.

About this task

You can define any number of WS-Notification service points for a given WS-Notification service. Each service point defined for the same WS-Notification service represents an alternative entry point to the service. Event notifications published to a particular WS-Notification service point are received by all applications connected to any service point of the same WS-Notification service (subject to subscription on the correct topic) regardless of the particular service point to which they are connected. For more information, see Reasons to create multiple WS-Notification service points.

When you create a Version 6.1 WS-Notification service point you select a bus member on which the WS-Notification service point is configured. You allocate a service point to a given bus member by specifying an endpoint listener that is configured for that bus member. You also choose the type of web service binding (SOAP over HTTP or SOAP over JMS) that is used for the WS-Notification service point.

The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

Procedure

1. Start the administrative console
2. Navigate to **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**. The “WS-Notification service points [Collection]” on page 3043 form is displayed. This form shows all the Version 6.1 WS-Notification service points configured for this Version 6.1 WS-Notification service.
3. In the content pane, click **New**. The New WS-Notification service point wizard is displayed. For more information about the properties that you set with the wizard, see “WS-Notification service points [Settings]” on page 3043.
4. Use the wizard to create the new Version 6.1 WS-Notification service point configuration by completing the following steps.
   a. Supply a name and (optional) description for the WS-Notification service point, and from the drop-down list select the bus member on which the service point is to be configured, then click **Next**. The service point name forms part of the URL used to access the service point (that is, the address of the web service that is exposed on the chosen server). On a single server system there is only one bus member in the list.
   b. Select a listener application to use to expose the service. Either select an existing endpoint listener for this bus member, or **Create a new endpoint listener**. For more information, see “Creating a new endpoint listener configuration” on page 2789.
5. Click **Finish**. If the processing completes successfully, the list of Version 6.1 WS-Notification service points for this Version 6.1 WS-Notification service is updated to include the new service point. Otherwise, an error message is displayed.
6. Save your changes to the master configuration.

What to do next

You can also use the administrative console to work with runtime information for service points. For more information, see “Interacting at run time with WS-Notification” on page 2982.

Modifying a Version 6.1 WS-Notification service point:

Modify an existing WS-Notification service point. A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

About this task

You can define any number of WS-Notification service points for a given WS-Notification service. Each service point defined for the same WS-Notification service represents an alternative entry point to the service. Event notifications published to a particular WS-Notification service point are received by all applications connected to any service point of the same WS-Notification service (subject to subscription on the correct topic) regardless of the particular service point to which they are connected. For more information, see Reasons to create multiple WS-Notification service points.

The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

To modify a WS-Notification service point use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**. The “WS-Notification service points [Collection]” on page 3043 form is displayed. This form shows all the WS-Notification service points configured for this WS-Notification service.
2. In the content pane, click the name of a WS-Notification service point in the list. The current settings for this WS-Notification service point are displayed in the “WS-Notification service points [Settings]” on page 3043 form.
3. Modify the general properties. The only general property that you can modify is the **description** property.
4. Modify the additional properties:
   a. Modify the administered subscribers that are associated with this WS-Notification service point. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time. For more information, see “Modifying a WS-Notification administered subscriber” on page 3010.
   b. Modify the custom properties, if any, that you have set for this WS-Notification service point. These custom properties are name and value pairs that you can use to set internal system configuration properties. In each pair, the name is a property key and the value is a string value.
   a. Modify the inbound ports that are associated with this WS-Notification service point.

      An inbound port describes the web service enablement of a service destination on a specific endpoint listener, with associated configuration. Each inbound port is associated with an endpoint listener, and you can control which groups of users can access a particular inbound service by making the service available only through specific endpoint listeners. For more information, see Endpoint listeners and inbound ports: Entry points to the service integration bus.

      You can use a JAX-RPC handler list to monitor activity at the port, and take appropriate action (for example logging, or rerouting) depending upon the sender and content of each message that passes through the port. For more information, see Bus-enabled web services and JAX-RPC handlers.

You can use WS-Security to set the levels of security to be applied to messages. The security level can be set independently for request and response messages. For more information, see Service integration technologies and WS-Security.

See also Inbound ports settings.

5. Apply any changes, then click **OK**. If the processing completes successfully, the list of WS-Notification service points for this WS-Notification service is redisplayed. Otherwise, an error message is displayed.
6. Save your changes to the master configuration.

What to do next

You can also use the administrative console to work with runtime information for service points. For more information, see "Interacting at run time with WS-Notification" on page 2982.

Deleting WS-Notification service points:

Delete one or more WS-Notification service points and the associated resources (Version 7.0 service point provider applications, or Version 6.1 inbound services and endpoint listeners).

Before you begin

Decide which method to use to configure these resources. You can delete a WS-Notification service point by using the administrative console as described in this task, or by using the "deleteWSNServicePoint command" on page 3064.

Do not use this task to delete service points as part of the process of deleting a WS-Notification service. When you delete a WS-Notification service, the associated service points and resources are automatically deleted.

About this task

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. When you delete a WS-Notification service point, the associated resources (Version 7.0 service point provider applications, or Version 6.1 inbound services and endpoint listeners) are automatically deleted. Do not delete these resources manually, because this might leave the associated configuration information in an inconsistent state.

Procedure

1. Start the administrative console.
2. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**. A list of all the WS-Notification service points is displayed in a "WS-Notification service points [Collection]" on page 3043 form.
3. Select the check box for every WS-Notification service point that you want to remove.
4. Click **Delete**. If the processing completes successfully, the list of WS-Notification service points for this WS-Notification service is updated. Otherwise, an error message is displayed.
5. Save your changes to the master configuration.

Creating a new WS-Notification administered subscriber:

As part of the configuration of a WS-Notification service point you can configure any number of administered subscribers for that service point. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

Before you begin

Decide which method to use to configure these resources. You can create a new administered subscriber by using the administrative console as described in this task, or by using the "createWSNAdministeredSubscriber command" on page 3068.

You should not define an administered subscriber for any of the endpoints exposed by the WS-Notification service on which it is being defined, because this would result in infinite looping of messages through the notification broker.

About this task

An administered subscriber contains the name of a NotificationProducer application or a (different) NotificationBroker endpoint and details of a subscription request (for example topic) that the WS-Notification service point should register as part of the server startup procedure. This enables you to pre-configure links between the NotificationBroker and a NotificationProducer, which can be a remote NotificationBroker or a NotificationProducer application.

To create a new administered subscriber, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers**. The "Administered subscribers [Collection]" on page 3033 form is displayed.
2. In the content pane, click **New**.
3. Specify the following properties for this administered subscriber.

   External web service endpoint
   The URL of the external web service to which the service should subscribe. That is, the endpoint reference (web address) of a notification producer or notification broker application. For example `http://remoteproducer.com`.

   Dialect
   The dialect in which the topic is expressed. The options are Simple, Concrete, or Full, as defined by the WS-Topics standard.

   Topic The topic on which the service should subscribe. This describes the class of notification messages that are delivered to the WS-Notification service point. For example `stock/IBM`. This property can include wildcards if they are supported by the topic dialect that you select.

   Topic namespace
   The URI that describes the topic namespace in which the specified topic is defined.

   Remote subscription timeout
   The length of time in hours after which the remote subscription will expire if not renewed by the server. This timeout minimizes the potential for orphaned subscriptions in the remote web service if the local server is uninstalled. Note that this field does not indicate the time at which the remote subscription is due to expire. Set the timeout length to something larger than the maximum length of time that the server is expected to remain offline, otherwise the stream of messages from the remote web service might be interrupted. While the server is running it

occasionally renews the remote subscription termination time (with the specified timeout) to prevent it from expiring during normal operation. If not specified, this timeout defaults to 24 (hours).

4. Click **OK**. If the processing completes successfully, the list of administered subscribers associated with the WS-Notification service point is updated to include the new administered subscriber. Otherwise, an error message is displayed.
5. Save your changes to the master configuration.

What to do next

You can also use the administrative console to list runtime information for administered subscribers. For more information, see "Listing active WS-Notification administered subscribers" on page 3031.

Modifying a WS-Notification administered subscriber:

As part of the configuration of a WS-Notification service point you can configure any number of administered subscribers for that service point. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

About this task

An administered subscriber contains the name of a NotificationProducer application or a (different) NotificationBroker endpoint and details of a subscription request (for example topic) that the WS-Notification service point should register as part of the server startup procedure. This enables you to pre-configure links between the NotificationBroker and a NotificationProducer, which can be a remote NotificationBroker or a NotificationProducer application.

To modify an administered subscriber, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers**. The "Administered subscribers [Collection]" on page 3033 form is displayed. This form shows all the administered subscribers configured for this WS-Notification service point.
2. In the content pane, click the name of an administered subscriber in the list. The current settings for this administered subscriber are displayed in the **Configuration** panel.
3. Modify the properties for this administered subscriber.

   External web service endpoint
   The URL of the external web service to which the service should subscribe. That is, the endpoint reference (web address) of a notification producer or notification broker application. For example `http://remoteproducer.com`.

   Dialect
   The dialect in which the topic is expressed. The options are Simple, Concrete, or Full, as defined by the WS-Topics standard.

   Topic The topic on which the service should subscribe. This describes the class of notification messages that are delivered to the WS-Notification service point. For example `stock/IBM`. This property can include wildcards if they are supported by the topic dialect that you select.

   Topic namespace
   The URI that describes the topic namespace in which the specified topic is defined.

Remote subscription timeout

The length of time in hours after which the remote subscription will expire if not renewed by the server. This timeout minimizes the potential for orphaned subscriptions in the remote web service if the local server is uninstalled. Note that this field does not indicate the time at which the remote subscription is due to expire. Set the timeout length to something larger than the maximum length of time that the server is expected to remain offline, otherwise the stream of messages from the remote web service might be interrupted. While the server is running it occasionally renews the remote subscription termination time (with the specified timeout) to prevent it from expiring during normal operation. If not specified, this timeout defaults to 24 (hours).

4. Apply any changes, then click **OK**. If the processing completes successfully, the list of administered subscribers associated with the WS-Notification service point is redisplayed. Otherwise, an error message is displayed.
5. Save your changes to the master configuration.

What to do next

You can also use the administrative console to list runtime information for administered subscribers. For more information, see “Listing active WS-Notification administered subscribers” on page 3031.

Deleting WS-Notification administered subscribers:

Delete one or more WS-Notification administered subscribers. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

Before you begin

Decide which method to use to configure these resources. You can delete an administered subscriber by using the administrative console as described in this task, or by using the “deleteWSNAdministeredSubscriber command” on page 3070.

About this task

To delete one or more administered subscribers, use the administrative console to complete the following steps :

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers**. A list of all the administered subscribers is displayed in an “Administered subscribers [Collection]” on page 3033 form.
2. Select the check box for every administered subscriber that you want to remove.
3. Click **Delete**. If the processing completes successfully, the list of administered subscribers for this WS-Notification service is updated. Otherwise, an error message is displayed.
4. Save your changes to the master configuration.

Creating a new WS-Notification permanent topic namespace:

Create a new permanent topic namespace. A topic namespace is a grouping of topics that allows information to be shared between applications. You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

Before you begin

Decide which method to use to configure these resources. You can create a new WS-Notification permanent topic namespace by using the administrative console as described in this task, or by using the “createWSNTopicNamespace command” on page 3074.

You can create many to many relationships between the set of permanent topic namespaces defined in a cell (that is for all WS-Notification services defined in that cell) and the service integration bus topic spaces with which they are associated. These relationships can become quite complex depending upon the topologies required by the applications that connect to the WS-Notification service. For guidance on when certain configurations might or might not be appropriate, see Options for associating a permanent topic namespace with a bus topic space.

About this task

A permanent topic namespace has the following characteristics:

- You can use it to expose an existing service integration bus topic space for use by WS-Notification clients, thus permitting interoperation between the WS-Notification applications and existing publish and subscribe applications connected to the bus such as JMS.
- You can use it to restrict the structure and content of the topic namespace by applying one or more topic namespace documents that describe the required structure.
- You can use it as part of a topic space mapping configured on a service integration bus link (between two service integration buses) or a topic mapping as part of a publish and subscribe bridge between a service integration bus and a WebSphere MQ network.

When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents.

You can also set a configuration attribute of a permanent topic namespace to control the reliability setting (persistence or non persistence) that is applied to any messages that use a given topic namespace.

To create a new WS-Notification permanent topic namespace, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces**. The “Permanent topic namespaces [Collection]” on page 3035 form is displayed.
2. In the content pane, click **New**.
3. Specify the following properties for this permanent WS-Notification topic namespace:
   a. Enter a name for the permanent topic namespace. The URI string by which this topic namespace is known. That is, the namespace URI by which WS-Notification applications refer to topics hosted by this namespace. For example `http://widgetproducer.com/prices`.
   b. Associate this new permanent topic namespace with the service integration bus topic space that you want to use to publish and receive messages. From the service integration bus topic space drop-down list, complete one of the following actions:
      - Select the name of an existing bus topic space.
      - Select the offered default name of *this_service_name*`TopicSpace` for a new bus topic space.
      - Select the option to `Create a new topic space`, then enter a name for the new topic space.

c. Select from the drop-down list the service integration bus reliability (quality of service) that is assigned to messages published through this topic namespace. You can choose one of five values. Each value represents one of the service integration bus message reliability levels. The default value is `reliable persistent`, which is the value used by default for JMS Persistent messages.

4. Click **OK**. If the processing completes successfully, the list of permanent WS-Notification topic namespaces is updated to include the new topic namespace. Otherwise, an error message is displayed.
5. Save your changes to the master configuration.

What to do next

To view the configuration of the associated service integration bus topic space, see "Showing the properties of a permanent WS-Notification topic namespace." To apply a topic namespace document, see "Applying a WS-Notification topic namespace document" on page 3015

Showing the properties of a permanent WS-Notification topic namespace:

A topic namespace is a grouping of topics that allows information to be shared between applications.You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

Before you begin

Decide which method to use to configure these resources. You can show the properties of a permanent WS-Notification topic namespace by using the administrative console as described in this task, or by using the "showWSNTopicNamespace command" on page 3078.

About this task

A permanent topic namespace has the following characteristics:

- You can use it to expose an existing service integration bus topic space for use by WS-Notification clients, thus permitting interoperation between the WS-Notification applications and existing publish and subscribe applications connected to the bus such as JMS.
- You can use it to restrict the structure and content of the topic namespace by applying one or more topic namespace documents that describe the required structure.
- You can use it as part of a topic space mapping configured on a service integration bus link (between two service integration buses) or a topic mapping as part of a publish and subscribe bridge between a service integration bus and a WebSphere MQ network.

When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents.

To show the properties of a permanent WS-Notification topic namespace, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces**. The "Permanent topic namespaces [Collection]" on page 3035 form is displayed. This form shows all the permanent topic namespaces configured for this WS-Notification service.

2. Optional: To view the configuration of the service integration bus topic space that is associated with a given permanent topic namespace, click the link in the **Service integration bus topic space** column. For more information, see “Topic space [Settings]” on page 2178
3. Optional: To view the configuration of the namespace documents that are associated with a given permanent topic namespace, click the link in the **Namespace documents** column.

   Note: This link also shows the number of namespace documents (0 or more) that are currently applied.

   The “Topic namespace document [Collection]” on page 3041 form is displayed. Use this form to apply, show or delete topic namespace documents.

Deleting WS-Notification permanent topic namespaces:

Delete topic namespace definitions from a WS-Notification service. A topic namespace is a grouping of topics that allows information to be shared between applications.You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

Before you begin

Decide which method to use to configure these resources. You can delete a WS-Notification permanent topic namespace using the administrative console as described in this task, or you can delete a WS-Notification permanent topic namespace using the wsadmin tool.

About this task

Deleting the topic namespace mapping that was used to establish a (currently active) subscription has the same effect as deleting the underlying service integration bus topic space, and subscriptions that were created using this namespace mapping are deleted. For more information about the effect that deleting a topic namespace has upon new and existing WS-Notification applications, see Failures as a result of changes in topic space and topic namespace configurations.

To delete one or more permanent topic namespaces, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces**. A list of all the permanent topic namespaces for this WS-Notification service is displayed in a “Permanent topic namespaces [Collection]” on page 3035 form.
2. Select the check box for every permanent topic namespace that you want to remove.
3. Click **Delete**.

   For each selected namespace, if the associated service integration bus topic space was created explicitly by the permanent topic namespace then you are asked whether or not to automatically delete the bus topic space. If the bus topic space was not created by the permanent topic namespace then you are not asked this question and the topic space is not deleted.

   Note: Deleting a service integration bus topic space causes exception messages to be generated for WS-Notification applications that reference the topic space, as described in Failures as a result of changes in topic space and topic namespace configurations.

   If the processing completes successfully, the list of permanent topic namespaces for this WS-Notification service is updated. Otherwise, an error message is displayed.
4. Save your changes to the master configuration.

Applying a WS-Notification topic namespace document:

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace. Use the administrative console to apply a topic namespace document to an existing topic namespace.

Before you begin

This task assumes that you have already created the topic namespace document that you want to apply.

Decide which method to use to configure these resources. You can apply a topic namespace document by using the administrative console as described in this task, or by using the “createWSNTopicDocument command” on page 3079.

About this task

For more information about the relationship between a permanent topic namespace and a topic namespace document, see Chapter 5 of the WS-Topics standard.

To apply a topic namespace document to an existing topic namespace, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents**. The “Topic namespace document [Collection]” on page 3041 form is displayed. This form shows all the topic namespace documents configured for this permanent topic namespace.
2. In the content pane, click **New**. The “Topic namespace document [Settings]” on page 3042 form is displayed.
3. Specify the following properties for this topic namespace document:
   a. URL of topic namespace document The URL of the topic namespace document that should be loaded.
   b. Optional: Description An optional description of the topic namespace document.
4. Click **OK**. If the processing completes successfully, the list of topic namespace documents for this permanent topic namespace is updated to include the new document. Otherwise, an error message is displayed.
5. Save your changes to the master configuration.

Showing the contents of a WS-Notification topic namespace document:

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace. Use the administrative console to show the contents of a topic namespace document.

Before you begin

Decide which method to use to configure these resources. You can show the contents of a topic namespace document by using the administrative console as described in this task, or by using the “showWSNTopicDocument command” on page 3083.

About this task

For more information about the relationship between a permanent topic namespace and a topic namespace document, see Chapter 5 of the WS-Topics standard.

To show the contents of a WS-Notification topic namespace document, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents**. The "Topic namespace document [Collection]" on page 3041 form is displayed. This form shows all the topic namespace documents configured for this permanent topic namespace.
2. In the content pane, click the name of a topic namespace document in the list. The contents of this topic namespace document are displayed.

Deleting WS-Notification topic namespace documents:

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace. Use the administrative console to delete one or more WS-Notification topic namespace documents.

Before you begin

Decide which method to use to configure these resources. You can delete a WS-Notification topic namespace document by using the administrative console as described in this task, or by using the "deleteWSNTopicDocument command" on page 3081.

About this task

To delete one or more WS-Notification topic namespace documents, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents**. A list of all the WS-Notification topic namespace documents is displayed in a "Topic namespace document [Collection]" on page 3041 form.
2. Select the check box for every WS-Notification topic namespace document that you want to remove.
3. Click **Delete**. If the processing completes successfully, the list of WS-Notification topic namespace documents for this WS-Notification topic namespace is updated. Otherwise, an error message is displayed.
4. Save your changes to the master configuration.

Publishing the WSDL files for a WS-Notification application to a compressed file:

Use the administrative console to download a compressed file with a `.zip` file extension that contains the published WSDL files for a WS-Notification application.

About this task

The ability to publish these WSDL files to a compressed file is particularly useful in the following circumstances:

- Writing a WS-Notification application that invokes web service operations against the NotificationBroker application, as described in Writing a WS-Notification application that does not expose a web service endpoint.
- Running the wsimport command against the exported `PublisherRegistrationManager.wsdl` file to generate a client stub for the PublisherRegistrationManager.
- Viewing the endpoint URLs to which WS-Notification applications connect, by looking in the WSDL file for the NotificationBroker application for Version 7.0 services, or the inbound service for Version 6.1 services.

Note:

When you run the wsimport command against the exported `PublisherRegistrationManager.wsdl` file you must include the `ibm-wsn-jaxws.xml` file as an argument to wsimport. If you omit this bindings file, the wsimport command fails with a naming conflict error concerning the ResourceNotDestroyedFault elements referred to in the `PublisherRegistrationManager.wsdl` file. For more information about why this exception occurs, see the following troubleshooting tip: The PublisherRegistrationManager.wsdl file is not successfully parsed by wsimport unless you include a JAX-WS bindings file.

The `ibm-wsn-jaxws.xml` file is located in the *app_server_root*`/util` directory. For example: `c:\was\util\ibm-wsn-jaxws.xml`. This bindings file expects to find the WSDL file to which it refers in the same directory as itself, so before you run the wsimport command you must copy the bindings file to the directory that holds your `PublisherRegistrationManager.wsdl` file. Here is an example of how to run the wsimport command to include the `ibm-wsn-jaxws.xml` file:

```
c:\was\bin\wsimport -b ibm-wsn-jaxws.xml -keep PublisherRegistrationManager.wsdl
```

Procedure

1. Start the administrative console.
2. Navigate to the “Publish WSDL files to .zip file [Settings]” form for the WS-Notification application.
   For JAX-WS based Version 7.0 WS-Notification services, click one of the following paths:
   - **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> [Additional Properties] Publish WSDL files to zip**
   - **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> [Additional Properties] Publish WSDL files to zip**

   For JAX-RPC based Version 6.1 WS-Notification services, click one of the following paths:
   - **Service integration -> WS-Notification -> Services -> *service_name* -> [Related Items] Notification broker inbound service settings** > **[Additional Properties] Publish WSDL files to ZIP file**
   - **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Related Items] Notification broker inbound service settings** > **[Additional Properties] Publish WSDL files to ZIP file**
3. Click on the file name to download a zip file that contains the application's published WSDL files.

Configuring JAX-WS handlers:

A JAX-WS handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. You can create JAX-WS

handlers, chain them together in the form of a handler list, then apply the handler list to a JAX-WS based Version 7.0 WS-Notification service point (for inbound invocation handling) or WS-Notification service (for outbound invocation handling).

About this task

The Java API for XML-based Web Services (JAX-WS) provides you with a standard way of developing interoperable and portable web services. To create a JAX-WS handler, you can use a tool such as IBM Rational Application Developer. To enable handlers to undertake more complex operations, you chain them together into handler lists. You associate each handler list with one or more JAX-WS based Version 7.0 WS-Notification services or service points, so that the handler list can monitor WS-Notification activity and take appropriate action depending upon the sender and content of each inbound or outbound message.

Detailed instructions on how to configure JAX-WS handlers and handler lists for use with JAX-WS based Version 7.0 WS-Notification services are provided in the following topics:

Procedure

- Load JAX-WS handler classes.
- Create a new JAX-WS handler configuration.
- Modify an existing JAX-WS handler configuration.
- Delete JAX-WS handler configurations.
- Create a new JAX-WS handler list.
- Modify an existing JAX-WS handler list.
- Delete JAX-WS handler lists.

Loading JAX-WS handler classes:

A JAX-WS handler interacts with messages through a JAX-WS based Version 7.0 WS-Notification service point (for inbound invocation handling) or WS-Notification service (for outbound invocation handling), therefore you must make the handler class available to the server or cluster that hosts the WS-Notification service point or service that you want to monitor.

Before you begin

This task assumes that you have already created your handler. You can do this by using IBM Rational Application Developer or a similar tool.

About this task

Before you can configure a JAX-WS handler for use with WS-Notification, you must make the handler class available to the server or cluster that hosts the WS-Notification service point or service that you want to monitor. To do this, you create a shared library for the class then add the shared library to the class loader for the server.

Procedure

1. Package the class file for your handler as a JAR file, then copy the JAR file into a convenient directory.

   Make the handler class available to the application server in one of the following ways:
   - Copy the individual class file into a directory structure under *app_server_root*/classes that matches the package name of the class, where *app_server_root* is the root directory for the installation of WebSphere Application Server. For example a handler class com.ibm.jaxws.handler.TestHandler is copied into the *app_server_root*/classes/com/ibm/jaxws/handler directory.
   - Package the class files for all your handlers as a JAR file, then copy it into the *app_server_root*/lib/app directory.

2. Start the administrative console.
3. Create a shared library for the JAR file.
   a. Navigate to **Environment -> Shared libraries**.
   b. Set the scope at which you want the new library to be visible, then click **New**.
   c. Give the new library a name.
   d. Set the class path to the directory and file name for your handler JAR file.
   e. Save your changes to the master configuration.

   For more information, see Creating shared libraries.
4. Create a class loader for the server on which you want to make the JAR file available.
   a. Navigate to **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server Infrastructure] Java and Process Management -> Class loader**.
   b. Click **New**.
   c. Click **OK**.
   d. Save your changes to the master configuration.

   For more information, see Configuring class loaders of a server.
5. Add the shared library to the class loader for the server.
   a. Navigate to **Servers -> Server Types -> WebSphere application servers -> *server_name* -> [Server Infrastructure] Java and Process Management -> Class loader -> *class_loader_name*** > **[Additional Properties] Shared library references**.
   b. Click **Add**.
   c. Click on the name of your new library, then click **OK**.
   d. Save your changes to the master configuration.

   For more information, see Associating shared libraries with servers.

What to do next

You are now ready to create a new JAX-WS handler configuration by using the administrative console or by using the “createJAXWSHandler command” on page 3086.

Creating a new JAX-WS handler configuration:

Create a Java API for XML-based Web Services (JAX-WS) handler configuration for use, as part of a handler list, with JAX-WS based Version 7.0 WS-Notification services.

Before you begin

You can create a new JAX-WS handler configuration by using the administrative console as described in this topic, or by using the “createJAXWSHandler command” on page 3086.

This task assumes that you have already created your handler. You can do this by using IBM Rational Application Developer or a similar tool. You must also make the handler class available to the server or cluster that hosts the WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling) that you want to monitor, as detailed in “Loading JAX-WS handler classes” on page 3018.

About this task

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To make WebSphere Application Server aware of your handler, and to make the handler available

for inclusion in one or more handler lists, you use the administrative console to create a new handler configuration.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> JAX-WS Handlers**. The JAX-WS handlers collection form is displayed.
3. Click **New**. The JAX-WS handlers settings form is displayed.
4. Type the following general properties:
   Name Type the name by which the handler is known.

   This name must be unique at cell scope, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.
   - It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

   For example `TestHandler`.

   Description
   Type the (optional) description of the handler.

   Class name
   Type the name of the class that is to be instantiated. This name must be a fully qualified java class name. For example `com.ibm.jaxws.handler.TestHandler`.

   Note: You can configure multiple instances of a handler by creating each instance with a different handler name, and pointing to the same handler class.
5. Click **OK**. The general properties for this item are saved, and the additional properties options are made available.
6. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handlers is updated to include the new handler. Otherwise, an error message is displayed.

What to do next

To use this handler, add it to a handler list as described in Creating a new JAX-WS handler list or Modifying an existing JAX-WS handler list.

Modifying an existing JAX-WS handler configuration:

Modify a Java API for XML-based Web Services (JAX-WS) handler configuration for a handler that is used, as part of a handler list, with JAX-WS based Version 7.0 WS-Notification services.

Before you begin

You can modify a JAX-WS handler configuration by using the administrative console as described in this topic, or by using the "modifyJAXWSHandler command" on page 3088.

If you modify a handler class but do not change the class name, you do not have to modify the handler configuration as described in this topic. You just have to stop then restart the servers or clusters that host the services or service points that this handler monitors.

About this task

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. You can use the administrative console to list existing handler configurations, and to view and modify their configuration details.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> JAX-WS Handlers**. A list of handlers is displayed in a JAX-WS handlers collection form.
3. Click the name of a handler in the list. The current JAX-WS handlers settings for this handler are displayed.
4. Modify the following general properties:
   Name Modify the name of the handler.

   This name must be unique at cell scope, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.
   - It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

   Note: When you change a handler name, the system looks up all objects that refer to it and updates the name.

   Description
   Modify the (optional) description of the handler.

   Class name
   Change the name of the class that is to be instantiated.

   If you change the class name, you must also make the new handler class available to the server or cluster that hosts the WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling) that you want to monitor, as detailed in "Loading JAX-WS handler classes" on page 3018.

   Note: You can configure multiple instances of a handler by creating each instance with a different handler name, and pointing to the same handler class.
5. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handlers is redisplayed. Otherwise, an error message is displayed.

Deleting JAX-WS handler configurations:

Delete the configuration for one or more Java API for XML-based Web Services (JAX-WS) handlers that are configured for use, as part of a handler list, with JAX-WS based Version 7.0 WS-Notification services.

Before you begin

You can delete a JAX-WS handler configuration by using the administrative console as described in this topic, or by using the "deleteJAXWSHandler command" on page 3089.

About this task

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request.

When you remove a handler that is currently used by one or more web services on a service integration bus, the system removes the handler from the handler lists for each associated web service.

You can use the administrative console to remove one or more handler configurations.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> JAX-WS Handlers**. A list of handlers is displayed in a JAX-WS handlers collection form.
3. Select the check box for every handler that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of handlers is updated. Otherwise, an error message is displayed.

Creating a new JAX-WS handler list:

Create a Java API for XML-based Web Services (JAX-WS) handler list for use with JAX-WS based Version 7.0 WS-Notification services.

Before you begin

You can create a new JAX-WS handler list by using the administrative console as described in this topic, or by using the “createJAXWSHandlerList command” on page 3092.

You can only add previously-configured handlers to a handler list. To configure a handler, see “Creating a new JAX-WS handler configuration” on page 3019.

About this task

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. The approach taken in WebSphere Application Server is to assign handler lists (rather than individual handlers) to WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling).

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> JAX-WS Handler Lists**. The JAX-WS handler lists collection form is displayed.
3. Click **New**. The JAX-WS handler lists settings form is displayed.
4. Type the following general properties:
   Name Type the name by which the handler list is known.

   This name must be unique at cell scope, and it must obey the following syntax rules:
   - It must not start with “.” (a period).

- It must not start or end with a space.
- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

For example TestList.

Description

Type the (optional) description of the handler list.

JAX-WS handlers

In the JAX-WS handlers pane, complete the following steps:

a. Select one or more handlers from the list of available JAX-WS handlers, then click **Add** to move the selected handlers into the list of handlers for this JAX-WS handler list.
b. Select a handler in the list of handlers for this JAX-WS handler list, then click **Up** or **Down** to change the position of the handler within the list.

Handlers are applied in the sequence in which they appear in the handler list.

Note: If you click **Reset**, only the **Name** and **Description** fields are reset to their state when the form was first loaded. The two lists of available and assigned handlers are not reset.

5. Click **OK**. The general properties for this item are saved.
6. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handler lists is updated to include the new handler list. Otherwise, an error message is displayed.

What to do next

To apply this handler list to inbound notification requests received at a Version 7.0 WS-Notification point, associate it with the service point as described in “Creating a new Version 7.0 WS-Notification service point” on page 3001 or “Modifying a Version 7.0 WS-Notification service point” on page 3003. To apply this handler list to outbound notification requests sent by your Version 7.0 WS-Notification service, associate the handler list with the service as described in “Creating a new Version 7.0 WS-Notification service” on page 2987 or “Modifying a Version 7.0 WS-Notification service” on page 2991.

For an overview of the end-to-end task of creating JAX-WS handlers, chaining them together in a handler list, then applying the handler list to a Version 7.0 WS-Notification service point or service, see “Applying a JAX-WS handler list to a WS-Notification service” on page 2973.

Modifying an existing JAX-WS handler list:

Modify the configuration details for a Java API for XML-based Web Services (JAX-WS) handler list that is configured for use with JAX-WS based Version 7.0 WS-Notification services.

Before you begin

You can modify a JAX-WS handler list by using the administrative console as described in this topic, or by using the “modifyJAXWSHandlerList command” on page 3093.

You can only add previously-configured handlers to a handler list. To configure a handler, see “Creating a new JAX-WS handler configuration” on page 3019.

About this task

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler

lists. The approach taken in WebSphere Application Server is to assign handler lists (rather than individual handlers) to WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling).

You can use the administrative console to list existing handler lists, and to view and modify their configuration details.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> JAX-WS Handler Lists**. A list of all the handler lists is displayed in a JAX-WS handler lists collection form.
3. Click the name of a handler list in the list. The current JAX-WS handler lists settings for this handler are displayed.
4. Modify the following general properties:
   Name Modify the name of the handler list.

   This name must be unique at cell scope, and it must obey the following syntax rules:
   - It must not start with "." (a period).
   - It must not start or end with a space.
   - It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

   When you change a handler list name, the system looks up all objects that refer to it and updates the name.

   Description
   Modify the (optional) description of the handler list.

   JAX-WS handlers
   In the JAX-WS handlers pane, complete the following steps:
   a. Select one or more previously-configured handlers from either the list of available JAX-WS handlers or the list of handlers for this JAX-WS handler list, then click **Add** or **Remove** to modify the list of handlers for this JAX-WS handler list.
   b. Select a handler in the list of handlers for this JAX-WS handler list, then click **Up** or **Down** to change the position of the handler within the list.

   Handlers are applied in the sequence in which they appear in the handler list.

   Note: If you click **Reset**, only the **Name** and **Description** fields are reset to their state when the form was first loaded. The two lists of available and assigned handlers are not reset.
5. Save your changes to the master configuration.

Results

If the processing completes successfully, the list of handler lists is redisplayed. Otherwise, an error message is displayed.

What to do next

To apply this handler list to inbound notification requests received at a Version 7.0 WS-Notification point, associate it with the service point as described in "Creating a new Version 7.0 WS-Notification service point" on page 3001 or "Modifying a Version 7.0 WS-Notification service point" on page 3003. To apply this handler list to outbound notification requests sent by your Version 7.0 WS-Notification service, associate the handler list with the service as described in "Creating a new Version 7.0 WS-Notification service" on page 2987 or "Modifying a Version 7.0 WS-Notification service" on page 2991.

For an overview of the end-to-end task of creating JAX-WS handlers, chaining them together in a handler list, then applying the handler list to a Version 7.0 WS-Notification service point or service, see "Applying a JAX-WS handler list to a WS-Notification service" on page 2973.

Deleting JAX-WS handler lists:

DeleteJava API for XML-based Web Services (JAX-WS) handler lists that are configured for use with JAX-WS based Version 7.0 WS-Notification services.

Before you begin

You can delete a JAX-WS handler list by using the administrative console as described in this topic, or by using the “deleteJAXWSHandlerList command” on page 3095.

About this task

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request.

When you remove a handler list that is currently used by one or more web services on a service integration bus, the system removes the handler list for each associated web service.

You can use the administrative console to remove one or more handler list configurations.

Procedure

1. Start the administrative console.
2. Navigate to **Service integration -> WS-Notification -> JAX-WS Handler Lists**. A list of handler lists is displayed in a JAX-WS handler lists collection form.
3. Select the check box for every handler list that you want to remove.
4. Click **Delete**.

Results

If the processing completes successfully, the list of handler lists is updated. Otherwise, an error message is displayed.

Interacting at run time with WS-Notification

View (and in some cases delete) at run time the active items associated with WS-Notification service points.

Before you begin

This task assumes that you have a fully configured and operational WS-Notification service point.

About this task

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service. The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

You can use the administrative console to interact at run time with the following active items associated with WS-Notification service points:

Publisher registrations
A runtime list of existing publisher registrations is provided. This includes, for each publisher registration, the information that was used to create it and when it will terminate.

Pull points
A runtime list is provided of the pull points that have been created. This includes, for each pull point, its current termination time and a link to the associated subscription.

Subscriptions
A runtime view is provided for subscriptions that have been created by applications. This includes, for each subscription, the information that was used to create it and an indication of its current state.

Administered subscribers
A runtime list is provided of the administered subscribers for a given WS-Notification service point. This includes, for each administered subscriber, an indication of its current state; for example whether the subscription was successfully initialized at start time.

For each WS-Notification service point, runtime information is available for subscriptions, registrations, pull points and administered subscribers. **For each WS-Notification service**, runtime information is available - aggregated for all service points for the service - for subscriptions, registrations and pull points. There is no aggregated view for administered subscribers at the WS-Notification service level.

To access and use the runtime information for active items associated with WS-Notification service points, use the administrative console to complete the following steps:

Procedure

1. Choose between information for a particular service point, or information aggregated for all service points for a particular service, by completing one of the following substeps:
   a. Optional: For runtime information for a particular service point, navigate to either **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations, pull points and administered subscribers for this WS-Notification service point.
   b. Optional: For runtime information aggregated for all service points for a particular service, navigate to either **Service integration -> WS-Notification -> Services -> *service_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name*** then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations and pull points for this WS-Notification service.
2. Click **Subscriptions** , **Publisher registrations**, **Pull points** or (for a particular service point) **Administered subscribers**.
   - If you click **Subscriptions**, a panel is displayed that lists the durable subscriptions that have been created by a WS-Notification service point in response to “Subscribe” requests from WS-Notification applications. You can view the subscription name (ID), topic and other information associated with the subscription, and you can view messages held on the durable subscription pending delivery. You can also delete subscriptions.

     Note: Both the WS-Notification subscription and publisher registration resources include the concept of scheduled termination, in which the application indicates a period of time after which the resource is destroyed. “Badly-behaved” in this case describes an application that requests an infinite lifetime for a resource and then does not explicitly delete the resource before it goes away (never to return).

- If you click **Publisher registrations**, a panel is displayed that lists the publisher registrations that are currently in effect on this WS-Notification service or service point (that is, applications that have registered as publishers). You can view the basic properties of the registration record. You can also delete a publisher registration record.
- If you click **Pull points**, a panel is displayed that lists the pull points that are currently active on this WS-Notification service or service point. You can view basic properties of the pull point such as the subscription with which it is associated and the time at which it is currently set to expire, and you can navigate to the associated subscriptions where appropriate. You can also delete pull points.
- For a particular service point, if you click **Administered subscribers**, a panel is displayed that lists the administered subscribers that are currently in effect on this WS-Notification service point. You can use this information to see whether a given subscriber has been successfully initialized.

What to do next

For more detailed information about working with individual runtime panels, see the following topics:
- "Listing or deleting active WS-Notification subscriptions."
- "Listing or deleting active WS-Notification publisher registrations" on page 3029.
- "Listing or deleting active WS-Notification pull points" on page 3030.
- "Listing active WS-Notification administered subscribers" on page 3031.

Listing or deleting active WS-Notification subscriptions:

List the subscriptions that exist at run time for a particular WS-Notification service point, or for all service points for a particular WS-Notification service.

About this task

The runtime panels for WS-Notification subscriptions list the durable subscriptions that have been created by a WS-Notification service point in response to "Subscribe" requests from WS-Notification applications. You can view the subscription name (ID), topic and other information associated with the subscription, and you can view messages held on the durable subscription pending delivery. You can also delete subscriptions. For example, to tidy up after a badly-behaved application.

Note: Both the WS-Notification subscription and publisher registration resources include the concept of scheduled termination, in which the application indicates a period of time after which the resource is destroyed. "Badly-behaved" in this case describes an application that requests an infinite lifetime for a resource and then does not explicitly delete the resource before it goes away (never to return).

A subscription created by a WS-Notification application can contain more than one topic expression, and each of these topic expressions can be in a different namespace, which can result in service integration bus subscriptions being created in multiple topic spaces.

To display a list of WS-Notification subscriptions that exist at run time, use the administrative console to complete the following steps:

Procedure

1. Choose between information for a particular service point, or information aggregated for all service points for a particular service, by completing one of the following substeps:
   a. Optional: For runtime information for a particular service point, navigate to either **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***, then click the **Runtime** tab. A panel is displayed

that contains links to runtime information about subscriptions, registrations, pull points and administered subscribers for this WS-Notification service point.

b. Optional: For runtime information aggregated for all service points for a particular service, navigate to either **Service integration -> WS-Notification -> Services -> *service_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name***, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations and pull points for this WS-Notification service.

2. Click **Subscriptions** The "Subscriptions [Collection]" on page 3040 form is displayed. This form shows all the currently active subscriptions for this WS-Notification service point or service. This collection form contains the following information about each list item:

Subscription id
The unique identifier of the subscription.

Topics
An array of topic names. Note that this will usually only contain one element.

Delivery state
The current runtime state of the subscription. In normal operation the subscription will show a state of **OK**. Other states are **PAUSED**, indicating that the application has paused the subscription as defined by the WS-Notification standards, or **ERROR**, indicating that the last attempt to deliver an event notification to the notification consumer was not successful. In the error case, more information about the cause of the failure can be found in the `SystemOut` log.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Consumer EPR
The endpoint reference to which event notifications matching the subscription are sent. That is, the endpoint reference to which notification messages are delivered as specified on the subscribe call.

Creation time
The time at which the subscription was created.

Termination time
The time at which the subscription will be deleted.

Pull type
Indicates whether the subscription is being used in pull mode. That is, whether this subscription is being accessed by a pull point. If the asynchronous (push) subscription model is being used, this field displays "No". If a pull point is being used, this field displays "Yes".

Service integration bus subscriptions
The service integration bus durable subscription or subscriptions that are associated with the WS-Notification subscription. This is a link to the service integration bus durable subscription detail panel for the subscriptions that contain data for this object.

3. Optional: To observe the runtime state of each of the service integration bus durable subscriptions that have been created, click the link in the **SIBus subscriptions** field.

If only one durable subscription has been created, the bus durable subscription detail panel is displayed. For more information, see "Administering durable subscriptions" on page 2002.

If multiple durable subscriptions have been created, for example if the WS-Notification subscription is spanned across multiple service integration bus topic spaces, a runtime collection form is displayed

that shows the individual bus durable subscriptions. This form shows all the currently active publisher registrations for this WS-Notification service point or service.

SIBus subscription
The service integration bus durable subscription ID. This links to the associated bus durable subscription detail panel. For more information, see “Administering durable subscriptions” on page 2002.

Topics
The list of topics that are contained within a single service integration bus topic space (and thus durable subscription).

What to do next

To delete one or more WS-Notification subscriptions, and the associated service integration bus durable subscriptions, select the check box next to each subscription name then click **Delete**.

Listing or deleting active WS-Notification publisher registrations:

List the publisher registrations that exist at run time for a particular WS-Notification service point, or for all service points for a particular WS-Notification service.

About this task

A WS-Notification application can register itself as a publisher in order to check that it is permitted to publish on the specified list of topics, or to initiate the demand based publisher pattern.

The runtime panels for WS-Notification publisher registrations list the publisher registrations that are currently in effect on this WS-Notification service or service point (that is, applications that have registered as publishers). You can view the basic properties of the registration record. You can also delete a publisher registration record. For example, to tidy up after a badly-behaved application.

Note: Both the WS-Notification subscription and publisher registration resources include the concept of scheduled termination, in which the application indicates a period of time after which the resource is destroyed. “Badly-behaved” in this case describes an application that requests an infinite lifetime for a resource and then does not explicitly delete the resource before it goes away (never to return).

To display a list of WS-Notification publisher registrations that exist at run time, use the administrative console to complete the following steps:

Procedure

1. Choose between information for a particular service point, or information aggregated for all service points for a particular service, by completing one of the following substeps:
   a. Optional: For runtime information for a particular service point, navigate to either **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations, pull points and administered subscribers for this WS-Notification service point.
   b. Optional: For runtime information aggregated for all service points for a particular service, navigate to either **Service integration -> WS-Notification -> Services -> *service_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name***, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations and pull points for this WS-Notification service.

2. Click **Publisher registrations** The "Publisher registrations [Collection]" on page 3038 form is displayed. This form shows all the currently active publisher registrations for this WS-Notification service point or service. This collection form contains the following information about each list item:

   Publisher ID
   The unique identifier of this publisher registration.

   Topic The topic on which the publisher is registered to publish. That is, the list of topics that are contained within a single service integration bus topic space (and thus durable subscription).

   Creation time
   The time at which the registration was created.

   Termination time
   The time at which the registration will be deleted.

   Demand based
   Indicates whether this is a demand based publisher. This displays "Yes" or "No".

   Producer EPR
   Additional publisher related data. The endpoint reference of the producer that created this registration.

What to do next

To delete one or more publisher registrations, select the check box next to each registration ID then click **Delete**.

Listing or deleting active WS-Notification pull points:

List the pull points that exist at run time for a particular WS-Notification service point, or for all service points for a particular WS-Notification service.

About this task

To use the synchronous delivery mechanism described by the WS-Notification standards, a WS-Notification application creates a pull point against the notification broker.

The runtime panels for WS-Notification pull points list the pull points that are currently active on this WS-Notification service or service point. You can view basic properties of the pull point such as the subscription with which it is associated and the time at which it is currently set to expire, and you can navigate to the associated subscriptions where appropriate. You can also delete pull points.

To display a list of WS-Notification pull points that exist at run time, use the administrative console to complete the following steps:

Procedure

1. Choose between information for a particular service point, or information aggregated for all service points for a particular service, by completing one of the following substeps:
   a. Optional: For runtime information for a particular service point, navigate to either **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations, pull points and administered subscribers for this WS-Notification service point.
   b. Optional: For runtime information aggregated for all service points for a particular service, navigate to either **Service integration -> WS-Notification -> Services -> *service_name*** or **Service**

integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name*, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations and pull points for this WS-Notification service.

2. Click **pull points** The "Pull points [Collection]" on page 3039 form is displayed. This form shows all the currently active pull points for this WS-Notification service point or service. This collection form contains the following information about each list item:

   Pull point id
   The unique identifier of the pull point.

   Creation time
   The time at which the pull point was created.

   Termination time
   The time at which the pull point will be deleted.

   Subscription id
   The unique identifier of the subscription associated with the pull point. If the pull point has not yet been supplied to a subscription, the text `Not Associated` is displayed.

What to do next

To delete one or more pull points, select the check box next to each pull point ID then click **Delete**.

Listing active WS-Notification administered subscribers:

List the administered subscribers that exist at run time for a particular WS-Notification service point. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

About this task

The runtime panels for WS-Notification administered subscribers list the administered subscribers that are currently in effect on this WS-Notification service point. You can use this information to see whether a given subscriber has been successfully initialized.

Administered Subscribers are defined on individual WS-Notification service points. There is no aggregated view for administered subscribers at the WS-Notification service level.

To display a list of WS-Notification administered subscribers that exist at run time, use the administrative console to complete the following steps:

Procedure

1. For runtime information for a particular service point, navigate to either **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name*** or **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***, then click the **Runtime** tab. A panel is displayed that contains links to runtime information about subscriptions, registrations, pull points and administered subscribers for this WS-Notification service point.
2. Click **Administered subscribers** The "Administered subscribers [Collection]" on page 3033 form is displayed. This form shows all the currently active administered subscribers for this WS-Notification service point. This collection form contains the following information about each list item:

   Producer EPR
   The endpoint reference of the remote web service application from which event notifications are received. That is, the endpoint reference (web address) of a notification producer or

notification broker application. For example `http://remoteproducer.com`. You provided this reference when you created the administered subscriber.

Topic The topic for which event notifications have been requested.

Subscription reference
The string form of the endpoint reference that was returned by the remote service as a result of the subscribe, if successfully created.

State The current runtime state of the administered subscriber. An icon indicator that displays either green or red to indicate whether the administered subscriber is OK or has failed. Move the mouse pointer over this icon to see a description of the current state. In the OK case, it describes the last successful operation - initially the time at which the subscription was contacted (for server startup) and subsequently the time the last message was received for this subscriber. In the failed case, it describes the details of the last failure.

Subscription timeout
The length of time in hours after which the remote subscription will expire if not renewed by the server. This timeout minimizes the potential for orphaned subscriptions in the remote web service if the local server is uninstalled. Note that this field does not indicate the time at which the remote subscription is due to expire. Set the timeout length to something larger than the maximum length of time that the server is expected to remain offline, otherwise the stream of messages from the remote web service might be interrupted. While the server is running it occasionally renews the remote subscription termination time (with the specified timeout) to prevent it from expiring during normal operation. If not specified, this timeout defaults to 24 (hours).

Configuring a JAX-WS client to resolve a WS-Notification service WSDL without following web links

You can configure a Java API for XML-based Web Services (JAX-WS) WS-Notification application (that is, a publisher, subscriber or consumer application) to resolve the WSDL parts for a Version 7.0 WS-Notification service from a local copy of the associated `jax-ws-catalog.xml` file and `wsdl` directory, rather than by following a series of web links.

About this task

When a JAX-WS WS-Notification client application gets the WSDL file from a Version 7.0 WS-Notification service, the client application expects to resolve the imported WSDL parts by following web links. However, if you copy the `jax-ws-catalog.xml` file and the `wsdl` directory for the WS-Notification service into the local file system for the client application, then the client application can resolve the imported WSDL parts from the local copy.

Procedure

1. Open a command prompt.
2. Navigate to directory *profile_root*`/config/cells/`*cell_name*`/buses/`*bus_name*`/wsn/`*wsn_service_name*`/META-INF`, where *profile_root* is the directory in which profile-specific information is stored.
3. Copy the `jax-ws-catalog.xml` file and the `wsdl` directory from this directory to the `WEB-INF` directory for your application.

Results

Your JAX-WS WS-Notification client application can resolve the WSDL parts for the WS-Notification service from the local copy of the catalog.

Administered subscribers [Collection]

An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

An administered subscriber contains the name of a NotificationProducer application or a (different) NotificationBroker endpoint and details of a subscription request (for example topic) that the WS-Notification service point should register as part of the server startup procedure. This enables you to pre-configure links between the NotificationBroker and a NotificationProducer, which can be a remote NotificationBroker or a NotificationProducer application.

External web service endpoint
The URL of the external web service to which the service should subscribe.

Dialect
The dialect in which the topic is expressed.

Topic The topic on which the service should subscribe.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Administered subscribers [Collection]

An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime > Administered subscribers**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime > Administered subscribers**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

External web service endpoint
The URL of the external web service to which the service should subscribe.

Topic The topic for which event notifications have been requested.

State The current runtime state of the administered subscriber.

Administered subscribers [Settings]

An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers -> *subscriber_name***
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Administered subscribers -> *subscriber_name***

An administered subscriber contains the name of a NotificationProducer application or a (different) NotificationBroker endpoint and details of a subscription request (for example topic) that the WS-Notification service point should register as part of the server startup procedure. This enables you to pre-configure links between the NotificationBroker and a NotificationProducer, which can be a remote NotificationBroker or a NotificationProducer application.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

External web service endpoint:

The URL of the external web service to which the service should subscribe.

That is, the endpoint reference (web address) of a notification producer or notification broker application. For example `http://remoteproducer.com`.

Required	Yes
Data type	String

Dialect:

The dialect in which the topic is expressed.

Required	Yes
Data type	drop-down list

Topic:

The topic on which the service should subscribe.

This describes the class of notification messages that are delivered to the WS-Notification service point. For example `stock/IBM`. This property can include wildcards if they are supported by the topic dialect that you select.

Required	Yes
Data type	String

Topic namespace:

The URI that describes the topic namespace in which the specified topic is defined.

Required	No
Data type	Custom

Remote subscription timeout:

The length of time in hours after which the remote subscription will expire if not renewed by the server.

This timeout minimizes the potential for orphaned subscriptions in the remote web service if the local server is uninstalled. Note that this field does not indicate the time at which the remote subscription is due to expire. Set the timeout length to something larger than the maximum length of time that the server is expected to remain offline, otherwise the stream of messages from the remote web service might be interrupted. While the server is running it occasionally renews the remote subscription termination time (with the specified timeout) to prevent it from expiring during normal operation. If not specified, this timeout defaults to 24 (hours).

Required	Yes
Data type	Integer
Range	1 through 2147483647

Permanent topic namespaces [Collection]

A topic namespace is a grouping of topics that allows information to be shared between applications.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

A permanent topic namespace has the following characteristics:

- You can use it to expose an existing service integration bus topic space for use by WS-Notification clients, thus permitting interoperation between the WS-Notification applications and existing publish and subscribe applications connected to the bus such as JMS.
- You can use it to restrict the structure and content of the topic namespace by applying one or more topic namespace documents that describe the required structure.
- You can use it as part of a topic space mapping configured on a service integration bus link (between two service integration buses) or a topic mapping as part of a publish and subscribe bridge between a service integration bus and a WebSphere MQ network.

When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents.

Namespace
The URI string by which this topic namespace is known.

Service integration bus topic space
The service integration bus topic space with which this namespace is associated.

Message reliability
The service integration bus reliability to apply to messages published to this topic namespace.

Topic namespace documents
The collection of topic namespace documents that define the structure of the topic namespace.

Buttons

New	Create a new administrative object of this type.
Delete	Delete the selected items.

Permanent topic namespace [Settings]

A topic namespace is a grouping of topics that allows information to be shared between applications.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces -> *namespace_name***
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces -> *namespace_name***

You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

A permanent topic namespace has the following characteristics:

- You can use it to expose an existing service integration bus topic space for use by WS-Notification clients, thus permitting interoperation between the WS-Notification applications and existing publish and subscribe applications connected to the bus such as JMS.
- You can use it to restrict the structure and content of the topic namespace by applying one or more topic namespace documents that describe the required structure.
- You can use it as part of a topic space mapping configured on a service integration bus link (between two service integration buses) or a topic mapping as part of a publish and subscribe bridge between a service integration bus and a WebSphere MQ network.

When you create a new WS-Notification permanent topic namespace, you specify the namespace and associate it with one of the service integration bus topic spaces configured on the bus on which the parent WS-Notification service is defined. You cannot modify a permanent topic namespace after it has been created, other than to apply or remove topic namespace documents.

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Namespace:

The URI string by which this topic namespace is known.

That is, the namespace URI by which WS-Notification applications refer to topics hosted by this namespace. For example `http://widgetproducer.com/prices`.

Required	Yes
Data type	Text

Service integration bus topic space:

The service integration bus topic space with which this namespace is associated.

That is, the bus topic space that is used by this topic namespace.

Required	Yes
Data type	Custom

Message reliability:

The service integration bus reliability to apply to messages published to this topic namespace.

Required	Yes
Data type	drop-down list

Range

Assured persistent
Messages are not discarded.

Reliable persistent
Messages might be discarded when a messaging engine fails.

Reliable non-persistent
Messages are discarded when a messaging engine stops or fails.

Express non-persistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable.

Best effort non-persistent
Messages are discarded when a messaging engine stops or fails. Messages might also be discarded if a connection used to send them becomes unavailable or as a result of constrained system resources.

Publisher registrations [Collection]

The set of applications that are currently registered as publishers with the notification broker.

To view this page in the console, click one of the following paths:

For information for a particular service point:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime** > **Publisher registrations**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime** > **Publisher registrations**

For information aggregated for all service points for a particular service:

- **Service integration -> WS-Notification -> Services -> *service_name*** > **Runtime** > **Publisher registrations**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Runtime** > **Publisher registrations**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Publisher ID
The unique identifier of this publisher registration.

Topic The topic on which the publisher is registered to publish.

Creation time
The time at which the registration was created.

Termination time
The time at which the registration will be deleted.

Demand based
Indicates whether this is a demand based publisher.

Producer EPR
Additional publisher related data.The endpoint reference of the producer that created this registration.

Buttons

Delete	Delete the selected items.

Pull points [Collection]

This collection lists the pull points that have been created on the associated bus member or members by WS-Notification applications. Use this panel to view the information about a pull point, such as the subscription with which it is associated and the time at which it is currently set to expire. You can also delete a pull point using the button provided.

To view this page in the console, click one of the following paths:

For information for a particular service point:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime** > **Pull points**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime** > **Pull points**

For information aggregated for all service points for a particular service:

- **Service integration -> WS-Notification -> Services -> *service_name* -> Runtime** > **Pull points**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Runtime** > **Pull points**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Pull point id
The unique identifier of the pull point.

Creation time
The time at which the pull point was created.

Termination time
The time at which the pull point will be deleted.

Subscription id
The unique identifier of the subscription associated with the pull point.

If the pull point has not yet been supplied to a subscription, the text `Not Associated` is displayed.

Buttons

Delete	Delete the selected items.

Service integration bus subscriptions [Collection]

This collection lists the service integration bus subscriptions associated with the previously selected WS-Notification subscription.

To view this pane in the console, complete the following steps:

1. Display “Subscriptions [Collection].” For example, click **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime** > **Subscriptions**
2. For a given WS-Notification service, click the link in the Service integration bus subscriptions column.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Subscription id
The unique identifier of the subscription.

Topic(s)
The topics on which the subscription is registered.

Subscriptions [Collection]

This collection lists the subscriptions that have been created on the associated bus member or members by WS-Notification applications. Use this panel to view the information about a subscription, such as the topic on which the subscription is registered and the time at which it is currently set to expire. You can also delete a subscription using the button provided.

To view this page in the console, click one of the following paths:

For information for a particular service point:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime** > **Subscriptions**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name* -> Runtime** > **Subscriptions**

For information aggregated for all service points for a particular service:

- **Service integration -> WS-Notification -> Services -> *service_name* -> Runtime** > **Subscriptions**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Runtime** > **Subscriptions**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Subscription id
The unique identifier of the subscription.

Topic(s)
The topics on which the subscription is registered.

Delivery state
The current runtime state of the subscription.

Consumer EPR
The endpoint reference to which event notifications matching the subscription are sent.

Creation time
The time at which the subscription was created.

Termination time
The time at which the subscription will be deleted.

Pull type
Indicates whether the subscription is being used in pull mode.

Service integration bus subscriptions
The service integration bus durable subscription or subscriptions that are associated with the WS-Notification subscription.

Buttons

Delete	Delete the selected items.

Topic namespace document [Collection]

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

Document name
The name of the file from which the topic namespace document was originally loaded.

Description
An optional description of the topic namespace document.

Buttons

New	Create a new administrative object of this type.

Delete	Delete the selected items.

Topic namespace document [Settings]

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents -> *document_name* > Upload**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents -> *document_name* > Upload**

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Associated permanent topic namespace:

The topic namespace to which this document is applied.

Required	No
Data type	String

URL of topic namespace document:

The URL of the topic namespace document that should be loaded.

Required	Yes
Data type	Text

Description:

An optional description of the topic namespace document.

Required	No
Data type	Text area

Topic namespace document [Settings]

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents -> *document_name* > View**

- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> Permanent topic namespaces -> *namespace_name* -> Topic namespace documents -> *document_name* > View**

WS-Notification service points [Collection]

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

You can define any number of WS-Notification service points for a given WS-Notification service. Each service point defined for the same WS-Notification service represents an alternative entry point to the service. Event notifications published to a particular WS-Notification service point are received by all applications connected to any service point of the same WS-Notification service (subject to subscription on the correct topic) regardless of the particular service point to which they are connected.

Name The name of the WS-Notification service point. This appears as part of the address of the web service that is exposed on the chosen server.

Associated bus member
The name of the bus member on which this WS-Notification service point is deployed.

Description
An optional description of the WS-Notification service point.

Buttons

New	Create a new service point.
Delete	Delete the selected items.
Copy	Copy the service point.

WS-Notification service points [Settings]

A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***

- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name* -> [Additional Properties] WS-Notification service points -> *point_name***

You can define any number of WS-Notification service points for a given WS-Notification service. Each service point defined for the same WS-Notification service represents an alternative entry point to the service. Event notifications published to a particular WS-Notification service point are received by all applications connected to any service point of the same WS-Notification service (subject to subscription on the correct topic) regardless of the particular service point to which they are connected.

The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

In this implementation of WS-Notification there are two types of WS-Notification service configuration:

- **Version 7.0**: Use this type of service if you want to compose a JAX-WS WS-Notification service with web service qualities of service (QoS) via policy sets, or if you want to apply JAX-WS handlers to your WS-Notification service. This is the recommended type of service for new deployments. This WS-Notification option has been available in WebSphere Application Server from Version 7.0.
- **Version 6.1**: Use this type of service if you want to expose a JAX-RPC WS-Notification service that uses the same technology provided in WebSphere Application Server Version 6.1, including the ability to apply JAX-RPC handlers to the service. This WS-Notification option has been available in WebSphere Application Server from Version 6.1.

Some of the fields are specific to the type of WS-Notification service that you are viewing.

- When you create a Version 6.1 WS-Notification service point you select a bus member on which the WS-Notification service point is configured. You allocate a service point to a given bus member by specifying an endpoint listener that is configured for that bus member. You also choose the type of web service binding (SOAP over HTTP or SOAP over JMS) that is used for the WS-Notification service point.
- When you create a Version 7.0 WS-Notification service point you select a bus member on which the WS-Notification service point is configured. You also choose the SOAP version that is supported by the service point, and (optionally) apply JAX-WS handler lists to the NotificationBroker, SubscriptionManager or PublisherRegistrationManager that are exposed through this service point.
- “Configuration tab”
- “Runtime tab” on page 3046

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the WS-Notification service point. This appears as part of the address of the web service that is exposed on the chosen server.

You cannot modify the name.

Required	Yes
Data type	String

Description:

An optional description of the WS-Notification service point.

Required	No
Data type	Text area

Associated bus member:

The name of the bus member on which this WS-Notification service point is deployed.

Required	No
Data type	String

General properties that are specific to Version 7.0 WS-Notification service points:

SOAP Version

Defines the version of SOAP supported by the service point. This affects the WSDL definition that will be exposed by the web service. Permitted values are 1.1 for SOAP 1.1 (the default), and 1.2 for SOAP 1.2.

Required	No
Data type	String

NotificationBroker JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the NotificationBroker endpoint of the WS-Notification service point.

Required	No
Data type	drop-down list

SubscriptionManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the SubscriptionManager endpoint of the WS-Notification service point.

Required	No
Data type	drop-down list

PublisherRegistrationManager JAX-WS handler list

The JAX-WS handler list that is applied to inbound requests from an application to the PublisherRegistrationManager endpoint of the WS-Notification service point.

Required	No
Data type	drop-down list

Additional Properties

Administered subscribers

An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

Custom properties

Select this link to configure additional custom properties for this WS-Notification service point.

Additional properties that are specific to Version 7.0 WS-Notification service points:

Policy set configuration
The policy set configuration associated with this WS-Notification service point. You can configure policy set and binding information for each port relating to this service point.

Publish WSDL files to zip
Publish the WSDL files for this service point to a compressed file.

Note: When you run the wsimport command against the exported `PublisherRegistrationManager.wsdl` file you must include the `ibm-wsn-jaxws.xml` file as an argument to wsimport.

For more information, see "Publishing the WSDL files for a WS-Notification application to a compressed file" on page 2984.

Service point application
The application associated with this WS-Notification service point. For Version 7.0 WS-Notification services, enterprise applications are used to expose the web services associated with the WS-Notification service.

Additional properties that are specific to Version 6.1 WS-Notification service points:

Notification broker inbound port settings
The inbound port defined for the notification broker role of this WS-Notification service point.

Publisher registration manager inbound port settings
The inbound port defined for the publisher registration manager role of this WS-Notification service point.

Subscription manager instance inbound port settings
The inbound port defined for the subscription manager role of this WS-Notification service point.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Additional Properties

Subscriptions
This collection lists the subscriptions that have been created on the associated bus member or members by WS-Notification applications. Use this panel to view the information about a subscription, such as the topic on which the subscription is registered and the time at which it is currently set to expire. You can also delete a subscription using the button provided.

Publisher registrations
The set of applications that are currently registered as publishers with the notification broker.

Pull points
This collection lists the pull points that have been created on the associated bus member or members by WS-Notification applications. Use this panel to view the information about a pull point, such as the subscription with which it is associated and the time at which it is currently set to expire. You can also delete a pull point using the button provided.

Administered subscribers
An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

WS-Notification services [Collection]

A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services**
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services**

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change which entries are listed, or to change the level of detail that is displayed for those entries, use the Filter settings.

A WS-Notification service provides the ability to expose some or all of the messaging resources defined on a service integration bus for use by WS-Notification applications.

Name The name of the WS-Notification service.

Service integration bus name
The name of the service integration bus with which the WS-Notification service is associated.

Type The type of the WS-Notification service.

- **Version 7.0**: Use this type of service if you want to compose a JAX-WS WS-Notification service with web service qualities of service (QoS) via policy sets, or if you want to apply JAX-WS handlers to your WS-Notification service. This is the recommended type of service for new deployments. This WS-Notification option has been available in WebSphere Application Server from Version 7.0.
- **Version 6.1**: Use this type of service if you want to expose a JAX-RPC WS-Notification service that uses the same technology provided in WebSphere Application Server Version 6.1, including the ability to apply JAX-RPC handlers to the service. This WS-Notification option has been available in WebSphere Application Server from Version 6.1.

Description
An optional description of the WS-Notification service.

Buttons

New	Create a new WS-Notification service.
Delete	Delete the selected items.

WS-Notification services [Settings]

A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

To view this page in the console, click one of the following paths:

- **Service integration -> WS-Notification -> Services -> *service_name***
- **Service integration -> Buses -> *bus_name* -> [Services] WS-Notification services -> *service_name***

A WS-Notification service provides the ability to expose some or all of the messaging resources defined on a service integration bus for use by WS-Notification applications.

In this implementation of WS-Notification there are two types of WS-Notification service configuration:

- **Version 7.0**: Use this type of service if you want to compose a JAX-WS WS-Notification service with web service qualities of service (QoS) via policy sets, or if you want to apply JAX-WS handlers to your WS-Notification service. This is the recommended type of service for new deployments. This WS-Notification option has been available in WebSphere Application Server from Version 7.0.
- **Version 6.1**: Use this type of service if you want to expose a JAX-RPC WS-Notification service that uses the same technology provided in WebSphere Application Server Version 6.1, including the ability to apply JAX-RPC handlers to the service. This WS-Notification option has been available in WebSphere Application Server from Version 6.1.

Some of the fields are specific to the type of WS-Notification service that you are viewing. For example, inbound service settings are specific to Version 6.1 WS-Notification services because Version 6.1 WS-Notification services use inbound services to expose the web services associated with the WS-Notification service. For Version 7.0 WS-Notification services, enterprise applications are used to expose the web services, and the configuration of these applications is accessed solely through the service point panels.

- "Configuration tab"
- "Runtime tab" on page 3052

Configuration tab

The Configuration tab shows configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Name:

The name of the WS-Notification service.

The name forms part of the endpoint on which the service is exposed (that is, the URL used to access the WS-Notification service points that are defined under the service). You cannot modify the name. For Version 6.1 WS-Notification services, the service name is unique within a bus. For Version 7.0 WS-Notification services the service name is unique within the cell, which matches the administration model used for policy sets and therefore supports composition of Version 7.0 WS-Notification services with WS-ReliableMessaging.

Required	Yes
Data type	String

Service integration bus name:

The name of the service integration bus with which the WS-Notification service is associated.

You cannot modify the service integration bus name.

Required	Yes
Data type	String

Type:

The type of the WS-Notification service.

- **Version 7.0**: Use this type of service if you want to compose a JAX-WS WS-Notification service with web service qualities of service (QoS) via policy sets, or if you want to apply JAX-WS handlers to your

WS-Notification service. This is the recommended type of service for new deployments. This WS-Notification option has been available in WebSphere Application Server from Version 7.0.

- **Version 6.1**: Use this type of service if you want to expose a JAX-RPC WS-Notification service that uses the same technology provided in WebSphere Application Server Version 6.1, including the ability to apply JAX-RPC handlers to the service. This WS-Notification option has been available in WebSphere Application Server from Version 6.1.

You choose the type when you create the service. You cannot modify the type for an existing service.

Required	Yes
Data type	String

Description:

An optional description of the WS-Notification service.

Required	No
Data type	Text area

Enable dynamic topic namespaces?:

Indicates whether dynamic topic namespaces can be used within the WS-Notification service.

That is, whether this service allows dynamic topic namespaces to be created at run time.

A dynamic topic namespace does not require manual administration by using the administration console or scripting. A dynamic topic namespace is used automatically in response to a request from a WS-Notification application for a topic namespace URI that has not been defined as a permanent topic namespace (assuming the WS-Notification service has been configured to permit use of dynamic namespaces).

A dynamic topic namespace has the following characteristics:

- It does not support interoperation between WS-Notification applications and other clients of the bus such as JMS.
- It is not possible to apply topic namespace documents to this topic space, and thus the structure and content of the topic space are unrestricted.
- It cannot be used as part of the configuration of service integration bus links or a publish and subscribe bridge.

Use this option to tightly control the topic namespaces that are used when connecting to a particular WS-Notification service (for example for security or auditing requirements). If you deselect this option, any applications that connect to the WS-Notification service and request topics from a dynamic topic namespace are stopped from publishing or receiving messages.

All messages published to a dynamic topic namespace are inserted with the default message reliability setting of reliable persistent. If this value is not acceptable, create a permanent topic namespace and manually configure the attribute to the appropriate value.

Note: The dynamic topic namespaces used on a particular WS-Notification service are backed by a service integration bus topic space that is created automatically when you create the topic namespace. The syntax of topics used within this topic space is internal to the WS-Notification service implementation.

Required	No

Data type Boolean

Dynamic topic space name:

The name of the service integration bus topic space to be used as the dynamic topic space for this WS-Notification service.

That is, the name of the bus topic space that is used to host the ad-hoc topic namespace, and to host dynamic topic namespaces if they are permitted. You cannot modify the dynamic topic space name.

Required No
Data type String

Requires registration:

Indicates whether publisher applications are required to register with the broker before they can publish notifications.

Required No
Data type Boolean

Fixed topic set:

Indicates whether the list of topics supported by the notification broker is fixed or may vary at runtime.

You cannot modify the setting for the fixed topic set.

Required No
Data type Boolean

Topic expression dialects:

List of topic dialects supported by this WS-Notification service.

That is, the name of the chosen topic dialect as defined by the WS-Topics standard:

- *Simple topics*. That is, single-level root topics with no wildcards. For example “stock”.
- *Concrete topics*. That is, multi-level topics with no wildcards. For example “stock/IBM”, “sport/football/results”.
- *Full topics*. That is, multi-level topics with wildcards and conjunctions. For example “stock//.”, “sport/football/*”, “sport/*/results”, “t1/t3 | t3/t4”.

You cannot modify the list of supported topic dialects.

Required Yes
Data type Text area

General properties that are specific to Version 7.0 WS-Notification services:

JAX-WS handler list

The JAX-WS handler list that is applied to outbound requests from the WS-Notification service.

A handler list defines the handlers that are applied when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume).

Required	No
Data type	drop-down list

Query WSDL

Indicates whether the Version 7.0 WS-Notification service queries the WSDL of other WS-Notification web services when interacting with them. By default, this option is enabled. By clearing this option, you can improve performance by avoiding expensive WSDL queries. However, you should note the following considerations when WSDL querying is not enabled:

- WS-Notification attempts to discover binding information (which is usually discovered through the WSDL) by using other means. WS-Notification uses the SOAP version associated with the WS-Notification service point where subscriptions were made (by other web services), or where administered subscriptions were created (by an administrator).
- There are some circumstances in which WS-Notification might be unable to determine binding information. This can happen when cleaning up subscriptions where the associated service point has been deleted and configuration information is no longer available. Under these circumstances WS-Notification makes a “best guess” at binding information to use to clean up the subscriptions.
- There is one scenario where incorrect binding information is used. That is, when a subscriber subscribes to use a particular SOAP binding, on behalf of a NotificationConsumer that expects notifications through a different SOAP binding.

Required	No
Data type	Boolean

General properties that are specific to Version 6.1 WS-Notification services:

JAX-RPC handler list

The JAX-RPC handler list that is applied to outbound requests from the WS-Notification service - for example the broker delivering notifications to a consumer.

A handler list defines the handlers that are applied when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume).

Required	No
Data type	drop-down list

Outbound security request binding

The security binding to be used with consumer notifications and remote publisher requests sent by this WS-Notification service.

WS-Security bindings define the security policy that is used when making outbound web service invocations, for example controlling demand-based publishers (subscribe, pause and resume).

Required	No
Data type	drop-down list

Outbound security response binding

The security binding to be used with remote publisher responses received by this WS-Notification service.

Required	No
Data type	drop-down list

Outbound security configuration

Specifies the details of how security is applied to requests and responses.

Required	No
Data type	drop-down list

Additional Properties

WS-Notification service points
Select this link to configure the deployment of WS-Notification service points on one or more servers.

Permanent topic namespaces
Select this link to configure permanent topic namespaces for the WS-Notification service.

Custom properties
Select this link to configure additional custom properties for this WS-Notification service.

Additional properties that are specific to Version 7.0 WS-Notification services:

Outbound request policy sets and bindings
The outbound request policy sets and bindings for the two WS-Notification service clients associated with this WS-Notification service. For reliable web service transmission of notification messages, use this option to associate the WS-Notification service client with a policy set that enables WS-ReliableMessaging.

Additional properties that are specific to Version 6.1 WS-Notification services:

Notification broker inbound service settings
The inbound service defined for the notification broker role of this WS-Notification service.

Publisher registration manager inbound service settings
The inbound service defined for the publisher registration manager role of this WS-Notification service.

Subscription manager inbound service settings
The inbound service defined for the subscription manager role of this WS-Notification service.

Runtime tab

The Runtime tab shows runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Additional Properties

Subscriptions
This collection lists the subscriptions that have been created on the associated bus member or members by WS-Notification applications. Use this panel to view the information about a subscription, such as the topic on which the subscription is registered and the time at which it is currently set to expire. You can also delete a subscription using the button provided.

Publisher registrations
The set of applications that are currently registered as publishers with the notification broker.

Pull points

This collection lists the pull points that have been created on the associated bus member or members by WS-Notification applications. Use this panel to view the information about a pull point, such as the subscription with which it is associated and the time at which it is currently set to expire. You can also delete a pull point using the button provided.

WSNotificationCommands command group for the AdminTask object

Decide which method to use to configure these resources. You can configure WS-Notification resources by using wsadmin command scripts as described in this topic, or you can configure WS-Notification resources by using the administrative console.

Although not part of the set of WS-Notification commands, the common task “Using a script to get up and running quickly with WS-Notification” on page 2964 also involves running a wsadmin command.

To run these commands, use the AdminTask object of the wsadmin scripting client. Each command acts on multiple objects in one operation. The commands are provided to allow you to make the most commonly-required types of update in a consistent manner, where modifying the underlying objects directly would be error-prone.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

The following administrative commands are available in this command group. Target objects and parameters are required unless otherwise stated.

- “createWSNService command” on page 3054
- “deleteWSNService command” on page 3057
- “listWSNServices command” on page 3058
- “showWSNService command” on page 3059
- “createWSNServicePoint command” on page 3061
- “deleteWSNServicePoint command” on page 3064
- “listWSNServicePoints command” on page 3065
- “showWSNServicePoint command” on page 3067
- “createWSNTopicNamespace command” on page 3074
- “deleteWSNTopicNamespace command” on page 3076
- “listWSNTopicNamespaces command” on page 3077
- “showWSNTopicNamespace command” on page 3078
- “createWSNAdministeredSubscriber command” on page 3068
- “deleteWSNAdministeredSubscriber command” on page 3070
- “listWSNAdministeredSubscribers command” on page 3071
- “showWSNAdministeredSubscriber command” on page 3072
- “createWSNTopicDocument command” on page 3079
- “deleteWSNTopicDocument command” on page 3081

- “listWSNTopicDocuments command” on page 3082
- “showWSNTopicDocument command” on page 3083
- “getWSN_SIBWSInboundService command” on page 3084
- “getWSN_SIBWSInboundPort command” on page 3085

createWSNService command

Use the createWSNService command to create a new WS-Notification service and the associated objects that form the infrastructure of the WS-Notification configuration. A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

You can create a new WS-Notification service by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Creating a new Version 6.1 WS-Notification service” on page 2995 and “Creating a new Version 7.0 WS-Notification service” on page 2987.

If you are creating a Version 6.1 WS-Notification service, you must first ensure that you have successfully configured an SDO repository as described in “Installing and configuring the SDO repository” on page 2772. The SDO repository is used to store WSDL documents during the creation of the Version 6.1 WS-Notification service. If you do not configure the repository, an error message is displayed when you create the service.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates a new WS-Notification service on a service integration bus. If you are creating a Version 6.1 WS-Notification service, the command also creates the three service integration bus inbound services that represent the three roles played by the broker, and creates the association between these objects and the new WS-Notification service.

Target object

An existing service integration bus (conditional - must be provided if the **bus** parameter is not supplied).

Required parameters

-name

The name to be given to the new WS-Notification service. The name forms part of the endpoint on which the service is exposed (that is, the URL used to access the WS-Notification service points that are defined under the service).

For Version 6.1 WS-Notification services, the service name is unique within a bus. For Version 7.0 WS-Notification services the service name is unique within the cell, which matches the administration model used for policy sets and therefore supports composition of Version 7.0 WS-Notification services with WS-ReliableMessaging.

Conditional parameters

-bus

The name of the service integration bus that is to host the WS-Notification service. This can be an existing bus, or the name of a new bus that you want the command to create for you. This parameter should only be specified if a Target object is not provided.

Optional parameters

-type

The type of WS-Notification service that is created. Permitted values are V7.0 and V6.1 (the default).

- **Version 7.0**: Use this type of service if you want to compose a JAX-WS WS-Notification service with web service qualities of service (QoS) via policy sets, or if you want to apply JAX-WS handlers to your WS-Notification service. This is the recommended type of service for new deployments. This WS-Notification option has been available in WebSphere Application Server from Version 7.0.
- **Version 6.1**: Use this type of service if you want to expose a JAX-RPC WS-Notification service that uses the same technology provided in WebSphere Application Server Version 6.1, including the ability to apply JAX-RPC handlers to the service. This WS-Notification option has been available in WebSphere Application Server from Version 6.1.

Only specify the following optional parameters if the service type is Version 6.1:

- **-jaxrpcHandlerList**
- **-outboundSecurityConfigName**
- **-outboundSecurityRequestBindingName**
- **-outboundSecurityResponseBindingName**

Note: For Version 7.0 WS-Notification services, equivalent functions to the Version 6.1 outbound security attributes are provided through policy set configuration.

Only specify the following optional parameters if the service type is Version 7.0:

- **-jaxwsHandlerListName**
- **-queryWSDL**

-description

An optional description of the WS-Notification service.

-permitsDynamicTopicNamespace

Indicates whether dynamic topic namespaces can be used within the WS-Notification service. That is, whether this service allows dynamic topic namespaces to be created at run time. For more information, see Dynamic topic namespace. Permitted values are TRUE (the default) and FALSE

Use this option to tightly control the topic namespaces that are used when connecting to a particular WS-Notification service (for example for security or auditing requirements). If you deselect this option, any applications that connect to the WS-Notification service and request topics from a dynamic topic namespace are stopped from publishing or receiving messages.

All messages published to a dynamic topic namespace are inserted with the default message reliability setting of reliable persistent. If this value is not acceptable, create a permanent topic namespace and manually configure the attribute to the appropriate value.

Note: The dynamic topic namespaces used on a particular WS-Notification service are backed by a service integration bus topic space that is created automatically when you create the topic namespace. The syntax of topics used within this topic space is internal to the WS-Notification service implementation.

-dynamicTopicSpace
The name of the service integration bus topic space to be used as the dynamic topic space for this WS-Notification service. That is, the name of the bus topic space that is used to host the ad-hoc topic namespace, and to host dynamic topic namespaces if they are permitted. If not specified, this value defaults to `WSN_dynamic_this_service_name`.

-requiresRegistration
Boolean flag. Indicates whether publisher applications are required to register with the broker before they can publish notifications. Permitted values are TRUE and FALSE (the default).

-jaxwsHandlerListName
The JAX-WS handler list that is applied to outbound requests from the WS-Notification service. A handler list defines the handlers that are applied when making outbound web service invocations, for example monitoring outbound event notification (in response to a subscribe operation) and controlling demand-based publishers (subscribe, pause and resume). For more information about handler lists, see “Configuring JAX-WS handlers” on page 2972.

Only specify this parameter for Version 7.0 WS-Notification services.

-jaxrpcHandlerListName
The JAX-RPC handler list that is applied to outbound requests from the WS-Notification service - for example the broker delivering notifications to a consumer. For more information about handler lists, see “Working with JAX-RPC handlers and clients” on page 2799.

Only specify this parameter for Version 6.1 WS-Notification services.

-outboundSecurityRequestBindingName
The security binding to be used with consumer notifications and remote publisher requests sent by this WS-Notification service.

Only specify this parameter for Version 6.1 WS-Notification services.

-outboundSecurityResponseBindingName
The security binding to be used with remote publisher responses received by this WS-Notification service.

Only specify this parameter for Version 6.1 WS-Notification services.

-outboundSecurityConfigName
Specifies the details of how security is applied to requests and responses.

Only specify this parameter for Version 6.1 WS-Notification services.

-queryWSDL
Boolean flag. Indicates whether the Version 7.0 WS-Notification service queries the WSDL of other WS-Notification web services when interacting with them. Permitted values are TRUE (the default) and FALSE.

By setting this parameter to FALSE you can improve performance by avoiding expensive WSDL queries. However, you should note the following considerations when WSDL querying is not enabled:

- WS-Notification attempts to discover binding information (which is usually discovered through the WSDL) by using other means. WS-Notification uses the SOAP version associated with the WS-Notification service point where subscriptions were made (by other web services), or where administered subscriptions were created (by an administrator).
- There are some circumstances in which WS-Notification might be unable to determine binding information. This can happen when cleaning up subscriptions where the associated service point

has been deleted and configuration information is no longer available. Under these circumstances WS-Notification makes a “best guess” at binding information to use to clean up the subscriptions.

- There is one scenario where incorrect binding information is used. That is, when a subscriber subscribes to use a particular SOAP binding, on behalf of a NotificationConsumer that expects notifications through a different SOAP binding.

Only specify this parameter for Version 7.0 WS-Notification services.

Examples

Create a Version 6.1 WS-Notification service (equivalent to omitting the **-type** parameter):

```
newService = AdminTask.createWSNService(["-bus", "bus1", "-name",
"NewWSNService", "-type", "V6.1"] )
```

Create a Version 7.0 WS-Notification service that allows composition with WS-ReliableMessaging:

```
newService = AdminTask.createWSNService(["-bus", "bus1", "-name",
"NewWSNService", "-type", "V7.0"] )
```

Create a Version 7.0 WS-Notification service with a non-null handler list:

```
newService = AdminTask.createWSNService(["-bus", "bus1", "-name",
"NewWSNService", "-type", "V7.0", "-jaxwsHandlerListName", "myHandlerList"] )
```

Create a Version 7.0 WS-Notification service that does not query WSDL:

```
newService = AdminTask.createWSNService(["-bus", "bus1", "-name",
"NewWSNService", "-type", "V7.0", "-queryWSDL", "false"] )
```

Set the custom property to enable strict topic checking on this WS-Notification service:

- Using Jython:

  ```
  propName = ["name", "com.ibm.ws.sib.wsn.strictTopicChecking"]
  propValue = ["value", "TRUE"]
  propAttrs = [propName, propValue]
  AdminConfig.create("Property", newService, propAttrs)
  ```

- Using Jacl:

  ```
  set propName [list name "com.ibm.ws.sib.wsn.strictTopicChecking"]
  set propValue [list value "TRUE"]
  set propAttrs [list $propName $propValue]
  $AdminConfig create Property $newService $propAttrs
  ```

deleteWSNService command

Use the deleteWSNService command to delete a WS-Notification service and associated resources. A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

You can delete a WS-Notification service by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting WS-Notification services” on page 2994.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes the specified WS-Notification service and all the objects configured on the service:

- For Version 7.0 WS-Notification services, the WS-Notification service points and the associated service point application are deleted.
- For Version 6.1 WS-Notification services, the WS-Notification service points and the associated service integration bus inbound ports are deleted.
- WS-Notification permanent topic namespaces and any service integration bus topic spaces that were created by the topic namespace are deleted.
- Topic namespace documents defined for the permanent topic namespaces, and (for Version 6.1 WS-Notification services) the associated XML documents stored in an SDO repository are deleted.

Target object

WSNService

Required parameters

None.

Conditional parameters

None.

Optional parameters

-deleteSIBTopicSpaces
TRUE or FALSE. Indicates whether service integration bus topic spaces created by definition of a WS-Notification permanent topic namespace should also be deleted (default is FALSE).

Example

Delete the WS-Notification service `newService` created in the example from topic “Creating a new WS-Notification service by using the wsadmin tool”.

- Using Jython:
  ```
  AdminTask.deleteWSNService(newService, ["-deleteSIBTopicSpaces", "TRUE"])
  ```
- Using Jacl:

```
$AdminTask deleteWSNService $newService {-deleteSIBTopicSpaces TRUE}
```

listWSNServices command

Use the listWSNServices command to list WS-Notification services. A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all the WS-Notification services in the configuration that match the specified input parameters. This command can be used to obtain a reference to one or more WS-Notification services that have already been created in the configuration in order to work with the service further, for example to add a new WS-Notification topic namespace definition.

Target object

WS-Notification services that match the requested pattern.

Required parameters

None.

Conditional parameters

None.

Optional parameters

-bus
The name of the service integration bus by which the list of WS-Notification services is filtered.

-name
The name of the WS-Notification service by which the list of WS-Notification services is filtered.

-type
The type of WS-Notification service by which the list of WS-Notification services is filtered. Permitted values are V7.0 and V6.1. By default, services of all types are listed.

Examples

List all WS-Notification services of any type on `bus1`:

```
wsnServiceList = AdminTask.listWSNServices(["-bus", "bus1"] )
```

List all Version 7.0 WS-Notification services on `bus1`:

```
wsnServiceList = AdminTask.listWSNServices(["-bus", "bus1", "-type", "V7.0"] )
```

showWSNService command

Use the showWSNService command to show the properties of a WS-Notification service. A WS-Notification service provides access to service integration bus resources for web services publish and subscribe clients.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command shows the properties of a WS-Notification service in a human-readable form. There are two patterns for use of this command:

- The required WS-Notification service is determined by the target object used.
- The required WS-Notification service is determined by the values specified on the parameters.

Target object

WSNService (conditional - must be provided if the **bus** and **name** parameters are not provided).

Required parameters

None.

Conditional parameters

-bus
The name of the service integration bus on which the WS-Notification service is located. This must be specified if a target object is not specified.

-name
The name of the WS-Notification service to be displayed. This must be specified if a target object is not specified.

Optional parameters

None.

Examples

Display the WS-Notification service obtained in the example from topic “Listing WS-Notification services by using the wsadmin tool”, by using the target object pattern.

- Using Jython:

  ```
  AdminTask.showWSNService(wsnServiceList)
  ```

- Using Jacl:

```
$AdminTask showWSNService $wsnServiceList
```

Display the WS-Notification service obtained in the example from topic “Listing WS-Notification services by using the wsadmin tool”, by using the parameters pattern.

- Using Jython:

  ```
  AdminTask.showWSNService(["-bus", "bus1", "-name", "NewWSNService"])
  ```

- Using Jacl:

```
$AdminTask showWSNService {-bus bus1 -name NewWSNService}
```

createWSNServicePoint command

Use the createWSNServicePoint command to create a new WS-Notification service point. A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

You can create a new WS-Notification service point by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Creating a new Version 6.1 WS-Notification service point” on page 3005 and “Creating a new Version 7.0 WS-Notification service point” on page 3001.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

This command creates the following resources:

- It creates a new WS-Notification service point on a WS-Notification service.
- For a Version 7.0 WS-Notification service point, it causes Web service endpoints for the three roles played by the broker to be exposed on the bus member.
- For a Version 6.1 WS-Notification service point, it creates the three service integration bus inbound ports that represent the three roles played by the broker, it creates a service integration bus endpoint listener if required, and it connects the new or existing endpoint listener to the service integration bus with which the WS-Notification service is associated.

If the service type is Version 7.0, do not specify the following optional parameters, which relate to JAX-RPC configuration:

-eplName
-eplURLRoot
-eplWSDLServingURLRoot

If the service type is Version 6.1, do not specify the following optional parameters, which relate to JAX-WS configuration:

-transportURLRoot
-transportSoapVersion
-jaxwsHandlerListNB
-jaxwsHandlerListSM
-jaxwsHandlerListPRM

You can only create service points for Version 7.0 WS-Notification services on WebSphere Application Server Version 7.0 or later bus members.

You can create service points for Version 6.1 WS-Notification services on WebSphere Application Server Version 6.1 or later application servers.

Target object

WSNService

Required parameters

-name
The name of the WS-Notification service point. This appears as part of the address of the web service that is exposed on the chosen server.

Conditional parameters

-server
Name of the server on which the WS-Notification service point is created.

- If you specify the **copyServicePoint** parameter, then you need not specify this parameter.
- If this parameter is used then the **node** parameter must be specified.

-node
Name of the node on which the server is located.

- If you specify the **copyServicePoint** parameter, then you need not specify this parameter.
- If this parameter is used then the **server** parameter must be specified.

For a Version 6.1 WS-Notification service, choose either to configure a new endpoint listener or use an existing one:

-eplName
Name of an endpoint listener.

- If you specify the **copyServicePoint** parameter, then you need not specify this parameter.
- If this endpoint listener has already been defined on the chosen server, then the **eplURLRoot** and **eplWSDLServingURLRoot** parameters should not be specified. Otherwise, the **eplURLRoot** and **eplWSDLServingURLRoot** parameters must both be specified.

Only specify this parameter for Version 6.1 WS-Notification services.

-eplURLRoot
Root of the externally visible endpoint address URL for web services that are accessed through this endpoint listener.

- If you specify the **copyServicePoint** parameter, then you need not specify this parameter.
- If the endpoint listener has already been defined on the chosen server, then this parameter should not be specified.

Only specify this parameter for Version 6.1 WS-Notification services.

-eplWSDLServingURLRoot
Root of the externally visible HTTP URL where the WSDL file associated with this endpoint listener is

located. In most circumstances this is `http://host_name:port_number/SIBWS`. For more information, see “Creating a new endpoint listener configuration” on page 2789 or “createSIBWSEndpointListener command” on page 2949.

- If you specify the **copyServicePoint** parameter, then you need not specify this parameter.
- If the endpoint listener has already been defined on the chosen server, then this parameter should not be specified.

Only specify this parameter for Version 6.1 WS-Notification services.

For a Version 7.0 WS-Notification service, configure the web service endpoint:

-transportURLRoot
Root of the externally visible endpoint address for the WS-Notification service point, in the following format:

protocol://*host_nameport_number/service_location*

where *protocol* is either `http` or `https`. For example:

```
http://myhostname:9080/ctx123/mySvc
```

You can use this parameter to associate a particular external web address with the WS-Notification service when you publish WSDL to a compressed file through the administrative console. Note that this address might not be the same as the address at which the WS-Notification service is exposed. This address is required when the WS-Notification service is accessed through a proxy.

If you do not specify this parameter, the underlying JAX-WS implementation creates an appropriate URL based on information provided as part of the service point installation process.

If you specify the **copyServicePoint** parameter, then you need not specify this parameter.

This attribute is the equivalent of the **eplURLRoot** parameter for service points created on Version 6.1 WS-Notification services. Only specify this parameter for Version 7.0 WS-Notification services.

-transportSoapVersion
Defines the version of SOAP supported by the service point. This affects the WSDL definition that will be exposed by the web service. Permitted values are 1.1 for SOAP 1.1 (the default), and 1.2 for SOAP 1.2.

If you specify the **copyServicePoint** parameter, then you need not specify this parameter.

Only specify this parameter for Version 7.0 WS-Notification services.

Copy the configuration from an existing WS-Notification service point definition:

-copyServicePoint
The name of an existing service point configured on the WS-Notification Service from which the other configuration attributes are copied. If you specify this parameter, then you only have to specify the **name** parameter. All other values are taken from the nominated existing service point.

Optional parameters

-description
An optional description of the WS-Notification service point.

-jaxwsHandlerListNB
The JAX-WS handler list that is applied to inbound requests from an application to the NotificationBroker endpoint of the WS-Notification service point.

If you specify the **copyServicePoint** parameter, then you need not specify this parameter.

Only specify this parameter for Version 7.0 WS-Notification services.

-jaxwsHandlerListSM
The JAX-WS handler list that is applied to inbound requests from an application to the SubscriptionManager endpoint of the WS-Notification service point.

If you specify the **copyServicePoint** parameter, then you need not specify this parameter.

Only specify this parameter for Version 7.0 WS-Notification services.

-jaxwsHandlerListPRM
The JAX-WS handler list that is applied to inbound requests from an application to the PublisherRegistrationManager endpoint of the WS-Notification service point.

If you specify the **copyServicePoint** parameter, then you need not specify this parameter.

Only specify this parameter for Version 7.0 WS-Notification services.

Examples

In the following examples, the WS-Notification service point uses the WS-Notification service `newService` created in the example from topic "Creating a new WS-Notification service by using the wsadmin tool".

Create a Version 6.1 WS-Notification service point on `server1` on `node1` and create a new endpoint listener that uses SOAP over HTTP on channel 1, where the host address of the server is `http://myHost:9080`:

```
newServicePoint = AdminTask.createWSNServicePoint(newService,
["-name", "newServicePoint", "-node", "node1", "-server", "server1", "-eplName", "myNewEPL",
"-eplURLRoot", "http://myhost:9080/wsn", "-eplWSDLServingURLRoot", "http://myhost:9080/sibws"] )
```

Create a Version 7.0 WS-Notification service point on `server1` on `node1` (minimum set of parameters):

```
newServicePoint = AdminTask.createWSNServicePoint(newService,
["-name", "newServicePoint", "-node", "node1", "-server", "server1",
"-transportURLRoot", "http://myhost:9080/myWSN"] )
```

Create a Version 7.0 WS-Notification service point on `server1` on `node1` (full set of parameters):

```
newServicePoint = AdminTask.createWSNServicePoint(newService,
["-name", "newServicePoint", "-node", "node1", "-server", "server1",
"-transportURLRoot", "http://myhost:9080/myWSN", "-transportSoapVersion", "1.1",
"-jaxwsHandlerListNB", "nbList", "-jaxwsHandlerListSM", "smList", "-jaxwsHandlerListPRM", "prmList"] )
```

deleteWSNServicePoint command

Use the deleteWSNServicePoint command to delete a WS-Notification service point and the associated resources. A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

You can delete a WS-Notification service point by using the wsadmin tool as described in this topic, or by using the administrative console as described in "Deleting WS-Notification service points" on page 3005.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

This command deletes the specified WS-Notification service point and all the objects configured on the service point:

- For Version 7.0 WS-Notification services, the associated service point application is deleted.
- For Version 6.1 WS-Notification services, the associated service integration bus inbound ports are deleted.
- For both versions, any associated WS-Notification administered subscribers are deleted.

Target object

WSNServicePoint and associated objects.

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

Delete the WS-Notification service point `newServicePoint`, created in the example from topic “Creating a new WS-Notification service point by using the wsadmin tool”.

- Using Jython:

```
AdminTask.deleteWSNServicePoint(newServicePoint)
```

- Using Jacl:

```
$AdminTask deleteWSNServicePoint $newServicePoint
```

listWSNServicePoints command

Use the listWSNServicePoints command to list the WS-Notification service points in the configuration of the target WS-Notification service that match the specified input parameters. A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

This command lists all the WSNServicePoint objects in the configuration of the target WSNService that match the specified input parameters. This command can be used to obtain a reference to one or more WSNServicePoint objects that have already been created in the configuration in order to work with the WSNServicePoint further - for example to add a new WSNTopicNamespace definition.

Target object

WSNServicePoint objects that match the requested pattern.

Required parameters

None.

Conditional parameters

None.

Optional parameters

-name
The name of the WSNServicePoint by which the list is filtered. Omitting this parameter results in the listing of all WSNServicePointsfor the target WSNService.

Example

Obtain a reference to the first WSNServicePoint defined against the wsnService object.

- Using Jython:

  ```
  wsnServicePointList = AdminTask.listWSNServicePoints(wsnService)
  wsnServicePoint = wsnServicePointList.split("\n")[0].rstrip()
  ```

- Using Jacl:

```
set wsnServicePointList [$AdminTask listWSNServicePoints $wsnService]
set wsnServicePoint [ lindex $wsnServicePointList 0 ]
```

showWSNServicePoint command

Use the showWSNServicePoint command to show the properties of a WS-Notification service point. A WS-Notification service point defines access to a WS-Notification service on a given bus member through a specified web service binding (for example SOAP over HTTP). Applications use the bus members associated with the WS-Notification service point to connect to the WS-Notification service.

Before you begin

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

The existence of a WS-Notification service point on a bus member implies that a WS-Notification web service is exposed from that bus member, and causes web service endpoints for the notification broker, subscription manager and publisher registration manager for this WS-Notification service to be exposed on the bus member with which the service point is associated. WS-Notification applications use these endpoints to interact with the WS-Notification service.

This command shows the properties of a WS-Notification service point in a human-readable form. There are two patterns for use of this command:

- The required service point is determined by the target service point.
- The required service point is determined by a combination of the target WS-Notification service, and the service point name provided as a parameter.

Target object

There are two choices for the target type of this command:

- WSNServicePoint (explicitly nominates the WS-Notification service point to be shown).
- WSNService (determines the required WS-Notification service; must be used in combination with the **name** parameter).

Required parameters

None.

Conditional parameters

-name

The name of the WS-Notification service point to be displayed. This must be specified if the target type is WSNService, and must not be specified if a WSNServicePoint target is supplied.

Optional parameters

None.

Examples

Show the properties of the WS-Notification service point `newServicePoint` created in the example from topic “Creating a new WS-Notification service point by using the wsadmin tool”:

- Using Jython:

```
AdminTask.showWSNServicePoint(newServicePoint)
```

- Using Jacl:

```
$AdminTask showWSNServicePoint $newServicePoint
```

Show the properties of the WS-Notification service point `newServicePoint`, created in the example from topic “Creating a new WS-Notification service point by using the wsadmin tool”, by using the WSNService target pattern:

- Using Jython:

```
AdminTask.showWSNServicePoint(newService, ["-name", "newServicePoint"])
```

- Using Jacl:

```
$AdminTask showWSNServicePoint $newService {-name newServicePoint}
```

createWSNAdministeredSubscriber command

Use the createWSNAdministeredSubscriber command to create a new administered subscriber. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

You can create a new WS-Notification administered subscriber by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Creating a new WS-Notification administered subscriber” on page 3008.

You should not define an administered subscriber for any of the endpoints exposed by the WS-Notification service on which it is being defined, because this would result in infinite looping of messages through the notification broker.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a new WS-Notification administered subscriber to the target WS-Notification service point.

Target Object

WSNServicePoint

Required parameters

-endpoint
The URL of the external web service to which the service should subscribe. That is, the endpoint reference (web address) of a notification producer or notification broker application. For example `http://remoteproducer.com`.

-topic
The topic on which the service should subscribe. This describes the class of notification messages that are delivered to the WS-Notification service point. For example `stock/IBM`. This property can include wildcards if they are supported by the topic dialect that you select.

-dialect
The dialect in which the topic is expressed. That is, the name of the chosen topic dialect as defined by the WS-Topics standard. Values of this parameter are SIMPLE, CONCRETE, FULL. For more information, see WS-Topics.

Conditional parameters

None.

Optional parameters

-topicNamespace
The URI that describes the topic namespace in which the specified topic is defined. Omitting this field indicates that the topic is contained in the ad-hoc topic namespace.

-remoteSubscriptionTimeout
The length of time in hours after which the remote subscription will expire if not renewed by the server. This timeout minimizes the potential for orphaned subscriptions in the remote web service if the local server is uninstalled. Note that this field does not indicate the time at which the remote subscription is due to expire. Set the timeout length to something larger than the maximum length of time that the server is expected to remain offline, otherwise the stream of messages from the remote web service might be interrupted. While the server is running it occasionally renews the remote subscription termination time (with the specified timeout) to prevent it from expiring during normal operation. If not specified, this timeout defaults to 24 (hours).

Example

Create an administered subscriber on the WS-Notification service point `newServicePoint` created in the example from topic “Creating a new WS-Notification service point by using the wsadmin tool”:

- Using Jython:

```
newAdminSub = AdminTask.createWSNAdministeredSubscriber(newServicePoint,
["-endpoint", "http://myremotehost:9080/producerEP", "-dialect", "SIMPLE",
 "-topic", "stock", "-topicNamespace", "http://example.org/mynamespace",
 "-remoteSubscriptionTimeout", 48] )
```

- Using Jacl:

```
set newAdminSub [ $AdminTask createWSNAdministeredSubscriber $newServicePoint
{ -endpoint http://myremotehost:9080/producerEP -dialect SIMPLE
 -topic stock -topicNamespace http://example.org/mynamespace
 -remoteSubscriptionTimeout 48 } ]
```

deleteWSNAdministeredSubscriber command

Use the deleteWSNAdministeredSubscriber command to delete a WS-Notification administered subscriber. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

You can delete a WS-Notification administered subscriber by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting WS-Notification administered subscribers” on page 3011.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes a WS-Notification administered subscriber.

Target object

WSNAdministeredSubscriber

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

Delete the administered subscriber `newAdminSub` created in the example from topic “Creating a new WS-Notification administered subscriber by using the wsadmin tool”:

- Using Jython:

  ```
  AdminTask.deleteWSNAdministeredSubscriber(newAdminSub)
  ```

- Using Jacl:

  ```
  $AdminTask deleteWSNAdministeredSubscriber $newAdminSub
  ```

listWSNAdministeredSubscribers command

Use the listWSNAdministeredSubscribers command to list the WS-Notification administered subscribers in the configuration of the target WS-Notification service point that match the specified input parameters. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all the administered subscribers in the configuration of the target WS-Notification service point that match the specified input parameters. You can use this command to obtain a reference to one or more administered subscribers that have already been created in the configuration in order to work with the administered subscriber further, for example to delete the definition.

Target object

WSNServicePoint

Required parameters

None.

Conditional parameters

None.

Optional parameters

-endpoint
The remote web service endpoint by which the list is filtered. That is, the endpoint reference (web address) of a notification producer or notification broker application. For example `http://remoteproducer.com`.

-topic
Topic expression describing the class of notification messages by which the list is filtered. This describes the class of notification messages that are delivered to the WS-Notification service point. For example `stock/IBM`. This property can include wildcards if they are supported by the topic dialect that you select.

-topicNamespace
The namespace URI by which the list is filtered.

-dialect
The dialect in which the topic is expressed. That is, the name of the chosen topic dialect as defined by the WS-Topics standard, by which the list is filtered. Values of this parameter are SIMPLE, CONCRETE, FULL. For more information, see WS-Topics.

Examples

Obtain a reference to the first administered subscriber defined against the `newServicePoint` object:

- Using Jython:

```
wsnSubscriberList = AdminTask.listWSNAdministeredSubscribers(newServicePoint)
wsnSubscriber = wsnSubscriberList.split("\n")[0].rstrip()
```

- Using Jacl:

```
set wsnSubscriberList [$AdminTask listWSNAdministeredSubscribers $newServicePoint]
set wsnSubscriber [ lindex $wsnSubscriberList 0 ]
```

Obtain a reference to the first administered subscriber defined against the `newServicePoint` object with a given topic:

- Using Jython:

```
wsnSubscriberList = AdminTask.listWSNAdministeredSubscribers(newServicePoint, ["-topic", "stock"])
wsnSubscriber = wsnSubscriberList.split("\n")[0].rstrip()
```

- Using Jacl:

```
set wsnSubscriberList [$AdminTask listWSNAdministeredSubscribers $newServicePoint {-topic stock}]
set wsnSubscriber [ lindex $wsnSubscriberList 0 ]
```

showWSNAdministeredSubscriber command

Use the showWSNAdministeredSubscriber command to show the properties of an administered subscriber. An administered subscriber provides a mechanism for the WS-Notification service point to subscribe to an external notification producer at server startup time.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command shows the properties of a WS-Notification administered subscriber object in a human-readable form. There are two patterns for use of this command:

- The required administered subscriber is determined by the target WSNAdministeredSubscriber object.
- The required administered subscriber is determined by a combination of the target WS-Notification service point, and the **endpoint**, **topic**, **topicNamespace** and **dialect** provided as parameters.

Target object

There are two choices for the target type of this command:

- WSNAdministeredSubscriber (explicitly nominates the administered subscriber to be shown).
- WSNServicePoint (determines the required WS-Notification service point; must be used in combination with the conditional parameters).

Required parameters

None.

Conditional parameters

-endpoint
The remote web service endpoint by which the list is filtered. That is, the endpoint reference (web address) of a notification producer or notification broker application. For example `http://remoteproducer.com`. This parameter must be specified if the target type is WSNServicePoint, and must not be specified if a WSNAdministeredSubscriber target is supplied.

-topic
Topic expression describing the class of notification messages by which the list is filtered. This describes the class of notification messages that are delivered to the WS-Notification service point. For example `stock/IBM`. This property can include wildcards if they are supported by the topic dialect that you select. This parameter must be specified if the target type is WSNServicePoint, and must not be specified if a WSNAdministeredSubscriber target is supplied.

-topicNamespace
The namespace URI by which the list is filtered. This parameter must be specified if the target type is WSNServicePoint, and must not be specified if a WSNAdministeredSubscriber target is supplied.

-dialect
The dialect in which the topic is expressed. That is, the name of the chosen topic dialect as defined by the WS-Topics standard, by which the list is filtered. Values of this parameter are SIMPLE, CONCRETE, FULL. For more information, see WS-Topics. This parameter must be specified if the target type is WSNServicePoint, and must not be specified if a WSNAdministeredSubscriber target is supplied.

Optional parameters

None.

Examples

Show the properties of the administered subscriber `newAdminSub` created in the example from topic "Creating a new WS-Notification administered subscriber by using the wsadmin tool":

- Using Jython:

```
AdminTask.showWSNAdministeredSubscriber(newAdminSub)
```

- Using Jacl:

```
$AdminTask showWSNAdministeredSubscriber $newAdminSub
```

Show the properties of the administered subscriber `newAdminSub`, created in the example from topic "Creating a new WS-Notification administered subscriber by using the wsadmin tool", by using the WSNServicePoint target pattern:

- Using Jython:

```
AdminTask.showWSNAdministeredSubscriber(newServicePoint, ["-topic", "stock"])
```

- Using Jacl:

```
$AdminTask showWSNAdministeredSubscriber $newServicePoint {-topic stock}
```

createWSNTopicNamespace command

Use the createWSNTopicNamespace command to create a new WS-Notification permanent topic namespace. A topic namespace is a grouping of topics that allows information to be shared between applications. You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

You can create a new WS-Notification permanent topic namespace by using the wsadmin tool as described in this topic, or by using the administrative console as described in "Creating a new WS-Notification permanent topic namespace" on page 3011.

You can create many to many relationships between the set of permanent topic namespaces defined in a cell (that is for all WS-Notification services defined in that cell) and the service integration bus topic spaces with which they are associated. These relationships can become quite complex depending upon the topologies required by the applications that connect to the WS-Notification service. For guidance on when certain configurations might or might not be appropriate, see Options for associating a permanent topic namespace with a bus topic space.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command creates the following resources:

- It creates a new WS-Notification topic namespace on a WS-Notification service and associates it with the specified service integration bus topic space.
- It creates the specified service integration bus topic space if it does not already exist.

You can also set a configuration attribute of a permanent topic namespace to control the reliability setting (persistence or non persistence) that is applied to any messages that use a given topic namespace.

Target Object

WSNService

Required parameters

-namespace
The URI string by which this topic namespace is known. That is, the namespace URI by which WS-Notification applications refer to topics hosted by this namespace. For example `http://widgetproducer.com/prices`.

-busTopicSpace
The service integration bus topic space with which this namespace is associated. That is, the bus topic space that is used by this topic namespace.

Conditional parameters

None.

Optional parameters

-reliability
The service integration bus reliability to apply to messages published to this topic namespace. Valid values for this property are as follows:

BEST_EFFORT_NONPERSISTENT

EXPRESS_NONPERSISTENT

RELIABLE_NONPERSISTENT

RELIABLE_PERSISTENT

ASSURED_PERSISTENT

Each value represents one of the service integration bus message reliability levels.

Examples

In the following examples, the WS-Notification topic namespace uses the WS-Notification service `newService` created in the example from topic “Creating a new WS-Notification service by using the wsadmin tool”.

Create a WS-Notification topic namespace on the WS-Notification service `newService`:

- Using Jython:

```
newTopicNamespace = AdminTask.createWSNTopicNamespace(newService,
["-namespace", "http://example.org/topicNamespace/example1",
"-busTopicSpace", "mySIBTopicspace"] )
```

- Using Jacl:

```
set newTopicNamespace [ $AdminTask createWSNTopicNamespace $newService
{ -namespace http://example.org/topicNamespace/example1
-busTopicSpace mySIBTopicspace } ]
```

Create a WS-Notification topic namespace on the WS-Notification service `newService` with a specific reliability:

- Using Jython:

```
newTopicNamespace = AdminTask.createWSNTopicNamespace(newService,
["-namespace", "http://example.org/topicNamespace/example1",
"-busTopicSpace", "mySIBTopicspace", "-reliability", "EXPRESS_NONPERSISTENT"] )
```

- Using Jacl:

```
set newTopicNamespace [ $AdminTask createWSNTopicNamespace $newService
{ -namespace http://example.org/topicNamespace/example1
-busTopicSpace mySIBTopicspace -reliability EXPRESS_NONPERSISTENT} ]
```

deleteWSNTopicNamespace command

Use the deleteWSNTopicNamespace command to delete a WS-Notification permanent topic namespace and the associated resources. A topic namespace is a grouping of topics that allows information to be shared between applications. You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

You can delete WS-Notification permanent topic namespaces by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting WS-Notification permanent topic namespaces” on page 3014.

Deleting the topic namespace mapping that was used to establish a (currently active) subscription has the same effect as deleting the underlying service integration bus topic space, and subscriptions that were created using this namespace mapping are deleted. For more information about the effect that deleting a topic namespace has upon new and existing WS-Notification applications, see Failures as a result of changes in topic space and topic namespace configurations.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command deletes all the objects associated with the specified WS-Notification topic namespace, including any linked topic namespace documents.

Target object

WSNTopicNamespace and associated objects.

Required parameters

None.

Conditional parameters

None.

Optional parameters

-deleteSIBTopicSpace
TRUE or FALSE. Indicates whether the associated service integration bus topic space is deleted with this object if the WS-Notification topic namespace was responsible for creating it. If the bus topic space existed before the WS-Notification topic namespace was created then this parameter has no effect and the bus topic space is not deleted.

Example

Delete the WS-Notification topic namespace `newTopicNamespace` created in the example from topic "Creating a new permanent WS-Notification topic namespace by using the wsadmin tool".

- Using Jython:

```
AdminTask.deleteWSNTopicNamespace(newTopicNamespace)
```

- Using Jacl:

```
$AdminTask deleteWSNTopicNamespace $newTopicNamespace
```

listWSNTopicNamespaces command

Use the listWSNTopicNamespaces command to list the WS-Notification topic namespaces in the configuration of the target WS-Notification service that match the specified input parameters. A topic namespace is a grouping of topics that allows information to be shared between applications.You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

This command lists all the WS-Notification topic namespaces in the configuration of the target WS-Notification service that match the specified input parameters. This command can be used to obtain a reference to one or more WS-Notification topic namespaces that have already been created in the configuration in order to work with the topic namespace further, for example to add a new topic namespace document definition.

Target Object

WSNService

Required parameters

None.

Conditional parameters

None.

Optional parameters

-namespace
The namespace URI of the WS-Notification topic namespace by which the list should be filtered. Omitting this parameter results in the listing of all WS-Notification topic namespaces for the target WS-Notification service.

Examples

Obtain a reference to the first WS-Notification topic namespace defined against the wsnService object:

- Using Jython:

```
AdminTask.listWSNTopicNamespaces(wsnService)
wsnNamespace = wsnNamespaceList.split("\n")[0].rstrip()
```

- Using Jacl:

```
set wsnNamespaceList [$AdminTask listWSNTopicNamespaces $wsnService]
set wsnNamespace [ lindex $wsnNamespaceList 0 ]
```

Obtain a reference to the WS-Notification topic namespace defined against the wsnService object with a given namespace:

- Using Jython:

```
wsnNamespaceList = AdminTask.listWSNTopicNamespaces(wsnService,
["-namespace", "http://example.org/topicNamespace/example1"] )
wsnNamespace = wsnNamespaceList.split("\n")[0].rstrip()
```

- Using Jacl:

```
set wsnNamespaceList [$AdminTask listWSNTopicNamespaces $wsnService
{-namespace http://example.org/topicNamespace/example1}]
set wsnNamespace [ lindex $wsnNamespaceList 0 ]
```

showWSNTopicNamespace command

Use the showWSNTopicNamespace command to show the properties of a WS-Notification topic namespace. A topic namespace is a grouping of topics that allows information to be shared between applications.You use a permanent topic namespace to statically define the association between a WS-Notification topic namespace URI and a service integration bus topic space destination.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

This command shows the properties of a WS-Notification topic namespace in a human-readable form. There are two patterns for use of this command:

- The required WS-Notification topic namespace is determined by the target WS-Notification topic namespace.
- The required WS-Notification topic namespace is determined by a combination of the target WS-Notification service, and the namespace of the WS-Notification topic namespace provided as a parameter.

Target object

There are two choices for the target type of this command:

- WSNTopicNamespace (explicitly nominates the WS-Notification topic namespace to be shown).
- WSNService (determines the required WS-Notification service; must be used in combination with the **namespace** parameter).

Required parameters

None.

Conditional parameters

-namespace
The namespace of the WS-Notification topic namespace to be displayed. This must be specified if the target type is WSNService, and must not be specified if a WSNTopicNamespace target is supplied.

Optional parameters

None.

Examples

Show the properties of the WS-Notification topic namespace `newTopicNamespace` created in the example from topic “Creating a new permanent WS-Notification topic namespace by using the wsadmin tool”:

- Using Jython:

```
AdminTask.showWSNTopicNamespace(newTopicNamespace)
```

- Using Jacl:

```
$AdminTask showWSNTopicNamespace $newTopicNamespace
```

Show the properties of the WS-Notification topic namespace `newTopicNamespace` created in the example from topic “Creating a new permanent WS-Notification topic namespace by using the wsadmin tool” using the WSNService target pattern:

- Using Jython:

```
AdminTask.showWSNTopicNamespace(newService,
 ["-namespace", "http://example.org/topicNamespace/example1"] )
```

- Using Jacl:

```
$AdminTask showWSNTopicNamespace $newService
{-namespace http://example.org/topicNamespace/example1 }
```

createWSNTopicDocument command

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace. Use the createWSNTopicDocument command to apply a topic namespace document to an existing topic namespace.

You can apply a WS-Notification topic namespace document to an existing topic namespace by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Applying a WS-Notification topic namespace document” on page 3015.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command adds a new WS-Notification topic namespace document to an existing WS-Notification topic namespace.

Target object

WSNTopicNamespace

Required parameters

-url
The URL of the topic namespace document that should be loaded. This URL must address a valid topic namespace document as defined in the WS-Topics Version 1.3 OASIS Standard

Conditional parameters

None.

Optional parameters

-description
An optional description of the topic namespace document.

Example

Create a new topic namespace document on the WS-Notification topic namespace `newTopicNamespace` created in the example from topic “Creating a new permanent WS-Notification topic namespace by using the wsadmin tool”:

- Using Jython:

  ```
  newTopicDoc = AdminTask.createWSNTopicDocument(newTopicNamespace,
  ["-url", "http://www.example.org/instance_doc1.xml"] )
  ```

- Using Jacl:

```
set newTopicDoc [ $AdminTask createWSNTopicDocument $newTopicNamespace
{ -url http://www.example.org/instance_doc1.xml } ]
```

Note that you must change the **url** parameter to point to a valid topic namespace document as defined in the WS-Topics Version 1.3 OASIS Standard.

deleteWSNTopicDocument command

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace. Use the deleteWSNTopicDocument command to delete a WS-Notification topic namespace document and the associated resources.

You can delete a WS-Notification topic namespace document by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting WS-Notification topic namespace documents” on page 3016.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

This command removes the XML file associated with the WS-Notification topic namespace document.

Target object

WSNTopicDocument and associated objects.

Required parameters

None.

Conditional parameters

None.

Optional parameters

None.

Example

Delete the topic namespace document `newTopicDoc` created in the example from topic “Applying a WS-Notification topic namespace document by using the wsadmin tool”:

- Using Jython:
  ```
  AdminTask.deleteWSNTopicDocument(newTopicDoc)
  ```
- Using Jacl:
  ```
  $AdminTask deleteWSNTopicDocument $newTopicDoc
  ```

listWSNTopicDocuments command

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace. Use the listWSNTopicDocuments command to list the topic namespace documents in the configuration of the target WS-Notification topic namespace that match the specified input parameters.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('WSNotificationCommands')
  ```
- For overview help on a given command, enter the following command at the wsadmin prompt:
  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command lists all the topic namespace documents in the configuration of the target WS-Notification topic namespace that match the specified input parameters. This command can be used to obtain a reference to one or more topic namespace documents that have already been created in the configuration in order to work with the topic namespace document further, for example to delete the definition.

Target object

WSNTopicNamespace documents that match the requested pattern.

Required parameters

None.

Conditional parameters

None.

Optional parameters

-url
The web address that was used to load the XML document.

Examples

Obtain a reference to the first topic namespace document defined against the `newTopicNamespace` object.

- Using Jython:

```
wsnDocList = AdminTask.listWSNTopicDocuments(newTopicNamespace)
wsnTopicDoc = wsnDocList.split("\n")[0].rstrip()
```

- Using Jacl:

```
set wsnDocList [$AdminTask listWSNTopicDocuments $newTopicNamespace]
set wsnTopicDoc [ lindex $wsnDocList 0 ]
```

Obtain a reference to the first topic namespace document defined against the `newTopicNamespace` object with a given URL.

- Using Jython:

```
wsnDocList = AdminTask.listWSNTopicDocuments(newTopicNamespace,
 ["-url", "http://www.example.org/instance_doc1.xml"])
wsnTopicDoc = wsnDocList.split("\n")[0].rstrip()
```

- Using Jacl:

```
set wsnDocList [$AdminTask listWSNTopicDocuments $newTopicNamespace
{-url http://www.example.org/instance_doc1.xml}]
set wsnDoc [ lindex $wsnDocList 0 ]
```

showWSNTopicDocument command

A topic namespace can optionally have topic namespace documents applied to it that define the structure of the topics that are permitted within the namespace. Use the showWSNTopicDocument command to show the contents of a WS-Notification topic namespace document.

You can show the contents of a WS-Notification topic namespace document by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Showing the contents of a WS-Notification topic namespace document” on page 3015.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

This command shows the XML contents of a WS-Notification topic namespace document. There are two patterns for use of this command:

- The required topic namespace document is determined by the target topic namespace document.
- The required topic namespace document is determined by a combination of the target WS-Notification topic namespace, and the **url** provided as a parameter.

Target object

There are two choices for the target type of this command:

- WSNTopicDocument (explicitly nominates the topic namespace document to be shown).
- WSNTopicNamespace (determines the required WS-Notification topic namespace; must be used in combination with the **url** parameter).

Required parameters

None.

Conditional parameters

-url
The web address that was used to load the XML document. This parameter must be specified if the target type is WSNTopicNamespace, and must not be specified if a WSNTopicDocument target is supplied.

Optional parameters

None.

Examples

Show the contents of the topic namespace document `newTopicDoc` created in the example from topic "Applying a WS-Notification topic namespace document by using the wsadmin tool":

- Using Jython:

```
AdminTask.showWSNTopicDocument(newTopicDoc)
```

- Using Jacl:

```
$AdminTask showWSNTopicDocument $newTopicDoc
```

Show the contents of the topic namespace document `newTopicDoc` created in the example from topic "Applying a WS-Notification topic namespace document by using the wsadmin tool", by using the WSNTopicNamespace target pattern:

- Using Jython:

```
AdminTask.showWSNTopicDocument(newTopicNamespace,
 ["-url", "http://www.example.org/instance_doc1.xml"])
```

- Using Jacl:

```
$AdminTask showWSNTopicDocument $newTopicNamespace
{ -url http://www.example.org/instance_doc1.xml }
```

getWSN_SIBWSInboundService command

Use the getWSN_SIBWSInboundService command to retrieve a reference to an inbound service that is associated with a Version 6.1 WS-Notification service.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

This command retrieves a reference to a specified service integration bus inbound service associated with a Version 6.1 WS-Notification service. Only use this command with Version 6.1 WS-Notification services. There are no bus-enabled inbound services associated with a Version 7.0 WS-Notification service.

Target object

WSNService

Required parameters

-type

The type of inbound service to retrieve. Valid options are:

- BROKER (notification broker)
- SUB_MGR (subscription manager)
- PUB_REG_MGR (publisher registration manager)

Conditional parameters

None.

Optional parameters

None.

Example

Retrieve a reference to the notification broker inbound service from the Version 6.1 WS-Notification service `newService` created in the example from topic “Creating a new WS-Notification service by using the wsadmin tool”:

```
brokerInboundService = AdminTask.getWSN_SIBWSInboundService(newService, ["-type", "BROKER"] )
```

getWSN_SIBWSInboundPort command

Use the getWSN_SIBWSInboundPort command to retrieve a reference to an inbound port associated with a Version 6.1 WS-Notification service point.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

This command retrieves a reference to a specified service integration bus inbound port associated with a Version 6.1 WS-Notification service point. Only use this command with Version 6.1 WS-Notification services. There are no bus-enabled inbound ports associated with a Version 7.0 WS-Notification service point..

Target object

WSNServicePoint

Required parameters

-type
The type of service integration bus inbound port to retrieve. Valid options are:

- BROKER (notification broker)
- SUB_MGR (subscription manager)
- PUB_REG_MGR (publisher registration manager)

Conditional parameters

None.

Optional parameters

None.

Example

Retrieve a reference to the subscription manager service integration bus inbound port from the Version 6.1 WS-Notification service point `newServicePoint` created in the example from topic “Creating a new WS-Notification service point by using the wsadmin tool”:

```
subManInboundPort = AdminTask.getWSN_SIBWSInboundPort(newServicePoint, ["-type", "SUB_MGR"] )
```

createJAXWSHandler command

Use the createJAXWSHandler command to create a new Java API for XML-based Web Services (JAX-WS) handler configuration so that the handler can be used, as part of a handler list, with Version 7.0 WS-Notification services.

You can create a new JAX-WS handler configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Creating a new JAX-WS handler configuration” on page 3019.

This task assumes that you have already created your handler. You can do this by using IBM Rational Application Developer or a similar tool. You must also make the handler class available to the server or cluster that hosts the WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling) that you want to monitor, as detailed in “Loading JAX-WS handler classes” on page 3018.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. This command creates a new JAX-WS handler configuration so that the handler can be used, as part of a handler list, with Version 7.0 WS-Notification services.

The configuration object associates a unique name (the **name** parameter) with a Java class (the **className** parameter) that refers to the JAX-WS handler implementation.

Target Object

A cell scope object.

Required parameters

-name
The name of the JAX-WS handler configuration object.

This name must be unique at cell scope, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: `\ / , # $ @ : ; " * ? < > | = + & % '`

-className
The name of the JAX-WS handler class that this configuration object represents. This name must be a fully qualified java class name. For example com.ibm.jaxws.handler.TestHandler.

Note: You can configure multiple instances of a handler by creating each instance with a different handler name, and pointing to the same handler class.

Conditional parameters

None.

Optional parameters

-description
A description of the JAX-WS handler.

Example

Create the configuration for a particular JAX-WS handler class:

- Using Jython:

```
targetCell = AdminConfig.list('Cell')
JAXWSHandler = AdminTask.createJAXWSHandler(targetCell,
["-name", "handler1", "-className", "handlerClass", "-description", "desc"])
```

- Using Jacl:

```
$AdminTask createJAXWSHandler targetCell
{-name handler1 -className handlerClass -description desc}
```

modifyJAXWSHandler command

Use the modifyJAXWSHandler command to modify a Java API for XML-based Web Services (JAX-WS) handler configuration for a handler that is used, as part of a handler list, with Version 7.0 WS-Notification services.

You can modify a JAX-WS handler configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Modifying an existing JAX-WS handler configuration” on page 3020.

If you modify a handler class but do not change the class name, you do not have to modify the handler configuration as described in this topic. You just have to stop then restart the servers or clusters that host the services or service points that this handler monitors.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. This command modifies a JAX-WS handler configuration that can be used, as part of a handler list, with Version 7.0 WS-Notification services.

The configuration object associates a unique name (the **name** parameter) with a Java class (the **className** parameter) that refers to the JAX-WS handler implementation.

Target Object

A JAX-WS handler configuration object.

Required parameters

-name
The name of the JAX-WS handler configuration object.

This name must be unique at cell scope, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

Note: When you change a handler name, the system looks up all objects that refer to it and updates the name.

-className

The name of the JAX-WS handler class that this configuration object represents. This name must be a fully qualified java class name. For example com.ibm.jaxws.handler.TestHandler.

If you change the class name, you must also make the new handler class available to the server or cluster that hosts the WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling) that you want to monitor, as detailed in "Loading JAX-WS handler classes" on page 3018.

Note: You can configure multiple instances of a handler by creating each instance with a different handler name, and pointing to the same handler class.

Conditional parameters

None.

Optional parameters

-description

A description of the JAX-WS handler.

Example

Modify the configuration for a particular JAX-WS handler class:

- Using Jython:

```
AdminTask.modifyJAXWSHandler(JAXWSHandler,
["-name", "newHandler1", "-className", "newHandlerClass",
 "-description", "newDesc"] )
```

- Using Jacl:

```
$AdminTask modifyJAXWSHandler JAXWSHandler
{-name newHandler1 -className newHandlerClass
 -description newDesc}
```

deleteJAXWSHandler command

Use the deleteJAXWSHandler command to delete the configuration for a Java API for XML-based Web Services (JAX-WS) handler that is configured for use, as part of a handler list, with Version 7.0 WS-Notification services.

You can delete a JAX-WS handler configuration by using the wsadmin tool as described in this topic, or by using the administrative console as described in "Deleting JAX-WS handler configurations" on page 3021.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('WSNotificationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. This command deletes the specified JAX-WS handler configuration that enables the handler to be used (as part of a handler list) with Version 7.0 WS-Notification services.

When you remove a handler that is currently used by one or more web services on a service integration bus, the system removes the handler from the handler lists for each associated web service.

Target object

The JAX-WS handler configuration object that is to be deleted.

Parameters

None.

Example

Delete the JAX-WS handler configuration object `JAXWSHandler`.

- Using Jython:
  `AdminTask.deleteJAXWSHandler(JAXWSHandler)`
- Using Jacl:

```
$AdminTask deleteJAXWSHandler JAXWSHandler
```

listJAXWSHandlers command

Use the listJAXWSHandlers command to list the Java API for XML-based Web Services (JAX-WS) handlers that are configured for use, as part of a handler list, with Version 7.0 WS-Notification services.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('WSNotificationCommands')`
- For overview help on a given command, enter the following command at the wsadmin prompt:
  `print AdminTask.help('command_name')`

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. This command lists all the JAX-WS handlers, for a given cell, that are configured for use, as part of a handler list, with Version 7.0 WS-Notification services.

Target Object

A cell scope object.

Parameters

None.

Example

List the JAX-WS Handler configuration objects for a given cell:

- Using Jython:

```
targetCell = AdminConfig.list('Cell')
AdminTask.listJAXWSHandlers(targetCell)
```

- Using Jacl:

```
$AdminTask listJAXWSHandlers targetCell
```

showJAXWSHandler command

Use the showJAXWSHandler command to show the properties of a Java API for XML-based Web Services (JAX-WS) handler that is configured for use (as part of a handler list) with Version 7.0 WS-Notification services.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. This command shows the properties of the specified JAX-WS handler configuration that enables the handler to be used (as part of a handler list) with Version 7.0 WS-Notification services.

Target object

A JAX-WS handler configuration object.

Parameters

None.

Example

Show the properties of the JAX-WS handler configuration object `JAXWSHandler`.

- Using Jython:

  ```
  AdminTask.showJAXWSHandler(JAXWSHandler)
  ```

- Using Jacl:

```
$AdminTask showJAXWSHandler JAXWSHandler
```

createJAXWSHandlerList command

Use the createJAXWSHandlerList command to create a Java API for XML-based Web Services (JAX-WS) handler list for use with Version 7.0 WS-Notification services.

You can create a new JAX-WS handler list by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Creating a new JAX-WS handler list” on page 3022.

You can only add previously-configured handlers to a handler list. To configure a handler, see the “createJAXWSHandler command” on page 3086.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. The approach taken in WebSphere Application Server is to assign handler lists (rather than individual handlers) to WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling).

Target Object

A cell scope object.

Required parameters

-name
The name of the JAX-WS handler list.

This name must be unique at cell scope, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

For example `TestList`.

Conditional parameters

None.

Optional parameters

-description
A description of the JAX-WS handler list.

-handlers
The list of JAX-WS handler names to be added to this list.

Each handler name supplied must exist as a JAX-WS handler object at cell scope. Handlers are applied in the sequence in which they appear in the handler list.

Example

Create the configuration for a particular JAX-WS handler list class:

- Using Jython:

```
targetCell = AdminConfig.list('Cell')
JAXWSHandlerList = AdminTask.createJAXWSHandlerList(targetCell,
'[-name handlerList1 -description desc -handlers [[handler1] [handler2]]]')
```

- Using Jacl:

```
$AdminTask createJAXWSHandlerList targetCell
{-name handlerList1 -description desc -handlers {{handler1}{handler2}} }
```

modifyJAXWSHandlerList command

Use the modifyJAXWSHandlerList command to modify the configuration details for a Java API for XML-based Web Services (JAX-WS) handler list that has been configured for use with Version 7.0 WS-Notification services.

You can modify a JAX-WS handler list by using the wsadmin tool as described in this topic, or by using the administrative console as described in "Modifying an existing JAX-WS handler list" on page 3023.

You can only add previously-configured handlers to a handler list. To configure a handler, see the "createJAXWSHandler command" on page 3086.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. The approach taken in WebSphere Application Server is to assign handler lists (rather than individual handlers) to WS-Notification service points (for inbound invocation handling) or WS-Notification services (for outbound invocation handling).

Target Object

A JAX-WS handler list object.

Required parameters

-name
The name of the JAX-WS handler list.

This name must be unique at cell scope, and it must obey the following syntax rules:

- It must not start with "." (a period).
- It must not start or end with a space.
- It must not contain any of the following characters: \ / , # $ @ : ; " * ? < > | = + & % '

When you change a handler list name, the system looks up all objects that refer to it and updates the name.

Conditional parameters

None.

Optional parameters

-description
A description of the JAX-WS handler list.

-handlers
The list of JAX-WS handler names to be added to this list.

Each handler name supplied must exist as a JAX-WS handler object at cell scope. Handlers are applied in the sequence in which they appear in the handler list.

Example

Modify the configuration for a particular JAX-WS handler list class:

- Using Jython:

```
JAXWSHandlerList = AdminTask.modifyJAXWSHandlerList(JAXWSHandlerList,
'[-name newHandlerList1 -description newDesc -handlers [[handler1] [handler2]]]')
```

- Using Jacl:

```
$AdminTask modifyJAXWSHandlerList JAXWSHandlerList
{-name handlerList1 -description newDesc -handlers {{handler1}{handler2}} }
```

deleteJAXWSHandlerList command

Use the deleteJAXWSHandlerList command to delete the configuration for a Java API for XML-based Web Services (JAX-WS) handler list that is configured for use with Version 7.0 WS-Notification services.

You can delete a JAX-WS handler list by using the wsadmin tool as described in this topic, or by using the administrative console as described in “Deleting JAX-WS handler lists” on page 3025.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

After using the command, save your changes to the master configuration; for example, by using the following command:

```
AdminConfig.save()
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. This command deletes the specified JAX-WS handler list.

When you remove a handler that is currently used by one or more web services on a service integration bus, the system removes the handler from the handler lists for each associated web service.

Target object

The JAX-WS handler list object that is to be deleted.

Parameters

None.

Example

Delete the JAX-WS handler list object `JAXWSHandlerList`:

- Using Jython:

  ```
  AdminTask.deleteJAXWSHandlerList(JAXWSHandlerList)
  ```

- Using Jacl:

```
$AdminTask deleteJAXWSHandlerList JAXWSHandlerList
```

listJAXWSHandlerLists command

Use the listJAXWSHandlerLists command to list the Java API for XML-based Web Services (JAX-WS) handler lists that are configured, for a given cell, for use with Version 7.0 WS-Notification services.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('WSNotificationCommands')
  ```

- For overview help on a given command, enter the following command at the wsadmin prompt:

  ```
  print AdminTask.help('command_name')
  ```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. This command lists all the JAX-WS handler lists for a given cell.

Target Object

A cell scope object.

Parameters

None.

Example

List the JAX-WS handler lists for a given cell:

- Using Jython:

  ```
  targetCell = AdminConfig.list('Cell')
  AdminTask.listJAXWSHandlerLists(targetCell)
  ```

- Using Jacl:

```
$AdminTask listJAXWSHandlerLists targetCell
```

showJAXWSHandlerList command

Use the showJAXWSHandlerList command to show the properties of a Java API for XML-based Web Services (JAX-WS) handler list that is configured for use with Version 7.0 WS-Notification services.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see Configuring Qshell to run WebSphere Application Server scripts using wsadmin scripting.

Command-line help is provided for service integration bus commands:

- For a list of the available WS-Notification commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('WSNotificationCommands')
```

- For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

Purpose

A Java API for XML-based Web Services (JAX-WS) handler is a Java class that performs a range of handling tasks. For example: logging messages, or transforming their contents, or terminating an incoming request. To enable handlers to undertake more complex operations, you chain them together into handler lists. This command shows the properties of the specified JAX-WS handler list.

Target object

A JAX-WS handler list object.

Parameters

None.

Example

Show the properties of the JAX-WS handler list `JAXWSHandlerList`:

- Using Jython:

```
AdminTask.showJAXWSHandlerList(JAXWSHandlerList)
```

- Using Jacl:

```
$AdminTask showJAXWSHandlerList JAXWSHandlerList
```

WS-Notification roles and goals

This topics lists a set of computing roles that members of your organization might perform, and explains how you can use WS-Notification to help meet the goals of each role.

For a general description of each of the following roles, see WebSphere Application Server roles and goals.

Enterprise architect

IT environments are currently evolving towards the following concepts:

- Service Oriented Architecture (SOA)
- Enterprise Service Bus (ESB)

The goal of the enterprise architect might be to guide their organization towards appropriate utilization of these concepts to maximize the efficiency and responsiveness of the business as a whole.

WS-Notification enables publish and subscribe communication patterns (such as a stock ticker) to be exposed by using web services in an SOA environment. This is done through open standards, enabling straightforward replacement of the service implementation. It promotes easy exchange of data between suppliers and customers through use of standard web service operations and prevents vendor lock-in or adoption of proprietary standards.

WebSphere Application Server also allows WS-Notification to be used as an on- or off-ramp to an ESB, providing seamless interchange of data between different types of client connected to the bus.

Solution architect

The main goal of the solution architect is to design a solution that supports the specification set by the enterprise architect. This might include providing an environment in which web service applications can participate in publish and subscribe messaging patterns. This participation might also include the requirement to be able to exchange event notifications between web service clients and other clients of the enterprise service bus.

To create a design, the solution architect completes the following broad steps:

- Learn about the support provided for WS-Notification in WebSphere Application Server.
- Select a hardware and software product combination for the enterprise that supports the WS-Notification standards.
- Design a server topology to host the applications, in accordance with the particular WS-Notification topologies that are to be implemented.

System administrator

For the specific steps that the system administrator performs to help implement common WS-Notification tasks, see the following topics:

- “Using a script to get up and running quickly with WS-Notification” on page 2964.
- “Configuring a WS-Notification service for use only by WS-Notification applications” on page 2968.
- “Providing access for WS-Notification applications to an existing bus topic space” on page 2969.
- “Securing WS-Notification” on page 2970.
- “Configuring JAX-WS handlers” on page 2972.
- “Applying a JAX-WS handler list to a WS-Notification service” on page 2973.
- “Configuring a Version 7.0 WS-Notification service with Web service QoS” on page 2974.
- “Configuring WS-Notification for reliable notification” on page 2976.
- “Migrating a Version 6.1 WS-Notification configuration from WebSphere Application Server Version 6.1 to Version 7.0 or later” on page 2977.
- “Preparing a migrated Version 6.1 WS-Notification configuration for reliable notification” on page 2978.
- “Interacting at run time with WS-Notification” on page 2982.
- “Publishing the WSDL files for a WS-Notification application to a compressed file” on page 2984.

Application developer

If the solution architect specifies a requirement to insert event notifications into the system (that is publish messages) or receive event notifications from the system as a result of creating a subscription containing an interest profile, then the application developer can use WS-Notification to meet this requirement.

There are various patterns of producing and consuming application defined by WS-Notification that are available for use by the application developer, depending upon the exact requirements of the application in question. These options are explored in the following common WS-Notification tasks:

- Writing a WS-Notification application that exposes a web service endpoint.
- Writing a WS-Notification application that does not expose a web service endpoint.

See also Developing applications that use WS-Notification and Filtering the message content of publications.

Chapter 33. Administering web services - Policy (WS-Policy)

WS-Policy is an interoperability standard that is used to describe and communicate the policies of a web service so that service providers can export policy requirements in a standard format. Clients can combine the service provider requirements with their own capabilities to establish the policies required for a specific interaction. This product conforms to the WS-Policy specification, so that policy information can be exchanged and received in accordance with the WS-Policy standard.

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. They implement a services oriented architecture (SOA), which supports the connecting or sharing of resources and data in a very flexible and standardized manner. Services are described and organized to support their dynamic, automated discovery and reuse.

Using WS-Policy to exchange policies in a standard format

WS-Policy is an interoperability standard that is used to describe and communicate the policies of a web service so that service providers can export policy requirements in a standard format. Clients can combine the service provider requirements with their own capabilities to establish the policies required for a specific interaction. WebSphere Application Server conforms to the WS-Policy specification, so that policy information can be exchanged and received in accordance with the WS-Policy standard.

About this task

For more information about using WS-Policy, see the following topics.

Procedure

- Configure a service provider to share its policy configuration
- Configure the client policy to use a service provider policy
- Configure the client policy to use a service provider policy from a registry
- Configure a service provider to share its policy configuration by using wsadmin scripting
- Configure the client policy based on a service provider policy by using wsadmin scripting
- Configure security for a WS-MetadataExchange request

Configuring a service provider to share its policy configuration

A WebSphere Application Server service provider can share its policy configuration in published Web Services Description Language (WSDL), or WSDL that is obtained by using an HTTP GET request or the Web Services Metadata Exchange (WS-MetadataExchange) GetMetadata request.

Before you begin

You have developed a web services service provider that contains all the necessary artifacts and deployed your web services application into your application server instance. You have attached the policy sets and managed the associated bindings.

For a list of WS-Policy assertion specifications and WS-Policy domains that are supported, see the topic about learning about WS-Policy.

About this task

You can make the policy configuration of a Java API for XML-Based Web Services (JAX-WS) service endpoint available to share in the following ways:

- Include the policy configuration of the service provider in the WSDL. The WSDL is then available to publish, or to obtain by using an HTTP GET request.

- Enable the Web Services Metadata Exchange (WS-MetadataExchange) protocol so that the policy configuration of the service provider is included in the WSDL and is available to a WS-MetadataExchange GetMetadata request. An advantage of using the WS-MetadataExchange protocol is that you can apply message-level security to WS-MetadataExchange GetMetadata requests by using a suitable system policy set.

If the service provider application uses multipart WSDL, all the WSDL must be local to the web service application. For more information about multipart WSDL, see the topic about WSDL.

You must configure a service provider to share its policy configuration because by default the policy configuration is not available in its WSDL. You can configure the service provider to include the policy configuration in its WSDL, to use WS-MetadataExchange so that the policy configuration is available, or both. This topic describes how to configure a service provider to share its policy configuration by using the administrative console. You can also configure a service provider to share its policy configuration by using wsadmin commands or Rational Application Developer tools.

You can configure a service provider to share its policy configuration at application or service level. The policy configuration that is represented by the policy sets attached to any lower levels will also be shared. Policy sets that are attached at lower levels override the policy set configuration attached at a higher level.

Procedure

1. From the navigation pane of the administrative console, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_provider_application_instance*** > **[Web services properties] Service provider policy sets and bindings**.
2. In the row for the application or service where the provider policy that you want to share is attached, click the link in the Policy sharing column. The link is either **Enabled** or **Disabled**. The Policy Sharing pane is displayed.
3. To include the policy configuration of the service provider in its WSDL so that it can be either published or obtained by using an HTTP GET request, select **Exported WSDL**.
4. To enable WS-MetadataExchange and make the policy configuration of the service provider available to a WS-MetadataExchange GetMetada request, select **WS-MetadataExchange request**.
5. Optional: If you select **WS-MetadataExchange request** and you want to use message-level security, select **Attach a system policy set to the WS-MetadataExchange**, then select a suitable policy set and binding from the drop-down lists. See “Configuring security for a WS-MetadataExchange request” on page 3115.
6. Click **OK** and save your changes to the master configuration.

Results

The policy configuration of the service provider is available to its clients. The WSDL of the service provider contains the current policy configuration in WS-PolicyAttachments format so that it is available to other clients, service registries, or services that support the Web Services Policy (WS-Policy) specification. The link in the Policy Sharing column on the Service provider policy sets and bindings pane changes to **Enabled**.

If the policy configuration cannot be shared, an error that describes the problem is written to the service provider error log, and the following policy is attached to the WSDL of the service provider:

```
<wsp:Policy>
<wsp:ExactlyOne>
</wsp:ExactlyOne>
</wsp:Policy>
```

This policy notifies the client that there is no acceptable policy configuration for the service. Other aspects of the WSDL are unaffected.

A service provider might not be able to share its policy configuration because the configuration cannot be expressed in the standard WS-PolicyAttachments format. One reason might be because multiple incompatible policies are defined for a particular attach point. Another reason might be because there is not enough binding information to generate the standard policy. Policy configuration might include bootstrap policy, for example, the policy to access a WS-Trust service, so the bootstrap policy must also be expressed in WS-PolicyAttachments format.

Configuring a service provider to share its policy configuration using wsadmin scripting

A WebSphere Application Server service provider can share its policy configuration in published Web Services Description Language (WSDL), or WSDL that is obtained by using an HTTP GET request or the Web Services Metadata Exchange (WS-MetadataExchange) GetMetadata request.

Before you begin

You have developed a web services service provider that contains all the necessary artifacts and deployed your web services application into your application server instance. You have attached the policy sets and managed the associated bindings.

For a list of WS-Policy assertion specifications and WS-Policy domains that are supported, see the topic about learning about WS-Policy.

About this task

You can make the policy configuration of a Java API for XML-Based Web Services (JAX-WS) service endpoint available to share in the following ways:

- Include the policy configuration of the service provider in the WSDL. The WSDL is then available to publish, or to obtain by using an HTTP GET request.
- Enable the Web Services Metadata Exchange (WS-MetadataExchange) protocol so that the policy configuration of the service provider is included in the WSDL and is available to a WS-MetadataExchange GetMetadata request. An advantage of using the WS-MetadataExchange protocol is that you can apply message-level security to WS-MetadataExchange GetMetadata requests by using a suitable system policy set.

If the service provider application uses multipart WSDL, all the WSDL must be local to the web service application. For more information about multipart WSDL, see the topic about WSDL.

You must configure a service provider to share its policy configuration because by default the policy configuration is not available in its WSDL. You can configure the service provider to include the policy configuration in its WSDL, to use WS-MetadataExchange so that the policy configuration is available, or both. This topic describes how to configure a service provider to share its policy configuration by using wsadmin commands. You can also use the administrative console or Rational Application Developer tools.

You can configure a service provider to share its policy configuration at application or service level. The policy configuration that is represented by the policy sets attached to any lower levels will also be shared. Policy sets that are attached at lower levels override the policy set configuration attached at a higher level.

Procedure

1. Start the wsadmin scripting client if it is not already running.
2. Use the SetProviderPolicySharingInfo command. For example:

```
AdminTask.setProviderPolicySharingInfo('[-applicationName WebServiceProviderApplication
-resource WebService:/WebServiceProvider.war:{http://example_path/}Service1
-sharePolicyMethods [httpGet ]]')
```

3. Save your changes to the master configuration.

   To save your configuration changes, enter the following command:

```
AdminConfig.save()
```

Results

The policy configuration of the service provider is available to its clients. The WSDL of the service provider contains the current policy configuration in WS-PolicyAttachments format so that it is available to other clients, service registries, or services that support the Web Services Policy (WS-Policy) specification.

If the policy configuration cannot be shared, an error that describes the problem is written to the service provider error log, and the following policy is attached to the WSDL of the service provider:

```
<wsp:Policy>
<wsp:ExactlyOne>
</wsp:ExactlyOne>
</wsp:Policy>
```

This policy notifies the client that there is no acceptable policy configuration for the service. Other aspects of the WSDL are unaffected.

A service provider might not be able to share its policy configuration because the configuration cannot be expressed in the standard WS-PolicyAttachments format. One reason might be because multiple incompatible policies are defined for a particular attach point. Another reason might be because there is not enough binding information to generate the standard policy. Policy configuration might include bootstrap policy, for example, the policy to access a WS-Trust service, so the bootstrap policy must also be expressed in WS-PolicyAttachments format.

What to do next

Optionally, you can publish the WSDL files.

setProviderPolicySharingInfo command:

Use the setProviderPolicySharingInfo command to set how an application or service that is a web service provider can share its policy configuration with other clients, service registries, or services that support the WS-Policy specification. You can set or remove this information about how a provider policy is shared.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available policy set management administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('PolicySetManagement')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the setProviderPolicySharingInfo command to set how an application, or a service in an application, shares its policy configuration with clients, service registries, or services that support the WS-Policy specification. The policy configuration is shared in WS-PolicyAttachments format.

The policy configuration of the resource can be shared with clients through a WS-MetadataExchange request, through Web Services Description Language (WSDL) exported by a ?WSDL HTTP Get request, or through both methods.

Target object

None.

Required parameters

-applicationName
The name of the application for which you want to set how the provider policy is shared. (String)

-resource
The name of the resource for which you want to set how the provider policy is shared. For all resources in an application, specify `WebService:/`. For a service in an application, specify `WebService:/`*`module`*`:{`*`namespace`*`}`*`service_name`*. Endpoints or operations inherit the settings of the parent application or service. (String)

Optional parameters

-sharePolicyMethods
Specifies how the policy configuration of the resource can be shared. (String array)

Enter either or both of the following values:

httpGet
The resource can share its policy configuration through WSDL that is obtained by a `?WSDL` HTTP Get request.

wsMex
The resource can share its policy configuration through a WS-MetadataExchange request.

-wsMexProperties
Specifies that message-level security is required for WS-MetadataExchange requests and specifies the settings that provide the message-level security. (Properties)

Enter the following values, following each value with the setting that you require for that value:

wsMexPolicySetName
The name of the system policy set that specifies message-level security when the resource shares its policy configuration through a WS-MetadataExchange request. Specify a system policy set that contains only WS-Security policies, only WS-Addressing policies, or both. The default policy set is SystemWSSecurityDefault.

wsMexPolicySetBinding
The name of the general binding for the policy set attachment when the resource shares its policy configuration through a WS-MetadataExchange request. Specify a general binding that is scoped to the global domain, or scoped to the security domain of this service. If you do not specify this property, the default binding is used.

This parameter is valid only when you specify wsMex for the **sharePolicyMethods** parameter.

-remove
Specifies whether the information about how the provider policy is shared is removed from the resource. (Boolean)

This parameter takes the following values:

true The information about how the provider policy is shared is removed from the resource.

false This value is the default. The information about how the provider policy is shared is not removed from the resource.

Examples

The following example removes the information about how the provider policy is shared from the WSSampleServices application:

```
AdminTask.setProviderPolicySharingInfo('[-applicationName WSSampleServices
-resource WebService:/ -remove true]')
```

The following example enables policy sharing, using WSDL exported by a `?WSDL` HTTP Get request, for the EchoService service in the WSSampleServices application:

```
AdminTask.setProviderPolicySharingInfo('[-applicationName WSSampleServices
-resource WebService:/WSSampleServicesSei.war:{http://example_path/}EchoService
-sharePolicyMethods [httpGet ]]')
```

The following example enables policy sharing, using a WS-MetadataExchange request with message-level security, for the WSSampleServices application. Message level security is provided by the SystemWSSecurityDefault policy set and the “Provider sample” general binding.

```
AdminTask.setProviderPolicySharingInfo('[-applicationName WSSampleServices
-resource WebService:/ -sharePolicyMethods [wsMex ]
-wsMexProperties [ [wsMexPolicySetName [SystemWSSecurityDefault]]
[wsMexPolicySetBinding [Provider sample]] ]]')
```

getProviderPolicySharingInfo command:

Use the getProviderPolicySharingInfo command to find out whether an application or service that is a web service provider can share its policy configuration, and list the properties that apply to sharing that configuration.

To run the command, use the AdminTask object of the wsadmin scripting client.

The wsadmin scripting client is run from Qshell. For more information, see the topic “Configure Qshell to run WebSphere Application Server scripts”.

This command is valid only when it is used with WebSphere Application Server Version 7 and later application servers. Do not use it with earlier versions.

For a list of the available policy set management administrative commands, plus a brief description of each command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('PolicySetManagement')
```

For overview help on a given command, enter the following command at the wsadmin prompt:

```
print AdminTask.help('command_name')
```

After using the command, save your changes to the master configuration. For example, use the following command:

```
AdminConfig.save()
```

Purpose

Use the getProviderPolicySharingInfo command to find out how a web services application, or a service in a Web services application, shares its policy configuration with clients, service registries, or services that support the WS-Policy specification. The policy configuration is shared in WS-PolicyAttachments format.

The command returns properties that show whether the policy configuration of the resource can be shared with clients through a WS-MetadataExchange request or through Web Services Description Language (WSDL) that is obtained by a `?WSDL` HTTP Get request.

Target object

None.

Required parameters

-applicationName
The name of the application for which you want to find out how it shares its policy configuration. The application must be a service provider. (String)

Optional parameters

-resource
The name of the resource for which you want to find out how it shares its policy configuration. If you specify this parameter, only the properties for that resource are returned. To retrieve information for the application, specify `WebService:/`. Alternatively, you can specify a service, endpoint or operation. However, policy sets are attached only at the application or service level, so the properties returned for an endpoint or operation are the settings that are inherited from the service. (String)

Return value

Returns a list of properties that include the resource name and that show whether the policy configuration of the resource can be shared. The following properties can be returned:

wsMexPolicySetName
The name of the policy set that specifies message-level security when the resource shares its policy configuration through a WS-MetadataExchange request. This property is returned if the value of the **sharePolicyMethods** property is wsMex and a policy set to provide message-level security was specified.

wsMexPolicySetBinding
The name of the binding that is applied when the resource shares its policy configuration through a WS-MetadataExchange request. This property is returned if the value of the **sharePolicyMethods** property is wsMex and a binding to provide message-level security was specified.

resource
The resource that you specified.

directSetting
How the properties apply to the resource. Valid values for this property are:

true
The properties apply directly to the resource.

false
The properties are inherited from the parent application or service.

sharePolicyMethods
How the policy configuration of the resource can be shared. Valid values for this property are:

httpGet
The resource shares its policy configuration through an HTTP Get request.

wsMex
The resource shares its policy configuration through a WS-MetadataExchange request.

Example

The following command displays the policy sharing configuration properties for the EchoService service in the WSSampleServices application. The provider is configured to share its policy through an HTTP Get request, and a WS-MetadataExchange request with message-level security. Message-level security for the WS-MetadataExchange request is provided by using the SystemWSSecurityDefault policy set and the "Provider sample" general binding.

```
AdminTask.getProviderPolicySharingInfo(['-applicationName', 'WSSampleServices',
'-resource', 'WebService:/SampleServicesSei.war:{http://example_path/}EchoService'])
.
.
[ [wsMexPolicySetName SystemWSSecurityDefault] [wsMexPolicySetBinding [Provider sample]]
[resource WebService:/SampleServicesSei.war:{http://example_path/}EchoService/]
[directSetting true] [sharePolicyMethods [httpGet wsMex]] ]
```

Policy sharing settings

Use this pane to view and change whether the policy configuration of a web services service provider is shared. You can configure the service provider to include the policy configuration in its Web Services Description Language (WSDL) so that it can be accessed using an HTTP Get request, or published. You can also make the policy configuration available to a Web Services Metadata Exchange (WS-MetadataExchange) request.

To view this administrative console page for an application or service, complete the following steps. The application must be a web services service provider.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_provider_application_name*** > **[Web Services Properties] Service provider policy sets and bindings**.
2. Select the link in the Policy Sharing column for the application or service you require. The link is available if the application or service has a policy set attached.

To view this administrative console page for a service, complete the following steps.

1. Click **Services** > **Service providers** > ***service_name***.
2. Select the link in the Policy Sharing column for the service. The link is available if the service has a policy set attached.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Exported WSDL:

Select **Exported WSDL** to include the policy configuration of the service provider in the WSDL. The policy configuration is in WS-PolicyAttachments format in the WSDL so that it is then available to other clients, service registries, or services that support the Web Services Policy (WS-Policy) specification. The policy configuration will be available in published WSDL, or a client can use an HTTP Get request that is targeted at the target URL followed by `?WSDL` to obtain the provider policy.

WS-MetadataExchange request:

Select **WS-MetadataExchange** to make the policy configuration of the service provider available to a WS-MetadataExchange GetMetadata request. The policy configuration is in WS-PolicyAttachments format

in the WSDL so that it is then available to other clients, service registries or services that support the Web Services Policy (WS-Policy) specification and the WS-MetadataExchange GetMetadata request.

When **WS-MetadataExchange** is selected, the **Attach a system policy set to the WS-MetadataExchange** option is available.

Attach a system policy set to the WS-MetadataExchange:

Select **Attach a system policy set to the WS-MetadataExchange** to set message-level security for the WS-MetadataExchange request. By default, this option is not selected and the transport policy of the application is used. This option is available only when **WS-MetadataExchange** is selected.

When **Attach a system policy set to the WS-MetadataExchange** is selected, the **Policy set** and **Binding** lists are available.

Policy set:

Select the policy set you require from the list to provide message-level security for the WS-MetadataExchange request. You can select from system policy sets that contain only WS-Security policies, only WS-Addressing policies, or both. The default policy set is SystemWSSecurityDefault.

System policy sets are used for system messages that are not business-related, for example, messages that apply qualities of service (QoS), including the messages that are defined in the WS-MetadataExchange protocol.

Note that any transport policy of the application is always used.

This option is available only when **Attach a system policy set to the WS-MetadataExchange** is selected.

Binding:

Select the binding you require from the list to provide message-level security for the WS-MetadataExchange request. You can select from general bindings that are scoped to the global domain, or scoped to the security domain of this service.

This option is available only when **Attach a system policy set to the WS-MetadataExchange** is selected.

Configuring the client policy to use a service provider policy

An application that is a web service client can obtain the policy configuration of a web service provider and use this information to establish a policy configuration that is acceptable to both the client and the service provider.

Before you begin

You have developed a web service client that contains all the necessary artifacts, and deployed your web services application into your application server instance. If you require them, you have attached the policy sets and managed the associated bindings.

The service provider must publish its policy in its Web Services Description Language (WSDL) and that policy must contain its policy configuration at run time in WS-PolicyAttachments format. The client must be able to support those provider policies.

For a list of WS-Policy assertion specifications and WS-Policy domains that are supported, see the WS-Policy topic.

About this task

You can administer the client to configure itself dynamically at run time, based on the policy of the service provider in the standard WS-PolicyAttachments format. You can administer the client to apply dynamically the provider policy at the application or service or service reference level. By default, endpoints and operations inherit their policy configuration from the relevant service. However, it is possible to configure a service reference to override the service, in which case the endpoints and operations inherit their policy configuration from the service reference.

If the provider policy uses multipart WSDL, you can use an HTTP GET request to obtain the policy of the provider, but you cannot use the WS-MetadataExchange protocol. For more information about multipart WSDL, see the topic about WSDL.

Policy intersection is the comparison of a client policy and a provider policy to determine whether they are compatible, and the calculation of a new policy, known as the effective policy, that complies with both their requirements and capabilities.

This topic describes how to configure the client policy to use a service provider policy by using the administrative console. You can also configure the client policy to use a service provider policy by using wsadmin commands.

Procedure

1. From the navigation panel of the administrative console, click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_client_application_instance*** > **[Web services properties] Service client policy sets and bindings**.
2. In the row for the application or service where you want to apply the policy, click the link in the Policies Applied column. The Policies Applied panel is displayed.
3. Select one of the following options from the drop-down list:
   - Provider policy only. Configure the client based solely on the policy of the service provider. This option is available when a client policy set is not attached.
   - Client and provider policy. Configure the client based on both the client policy set and the policy of the service provider. This option is available when a client policy set is attached.

   The other options in the list do not apply to this task.
4. Use the radio buttons to select which method to employ to obtain the provider policy: an HTTP GET request (see step 5) or a WS-MetadataExchange request (see step 6).
5. Optional: To obtain the provider policy by using an HTTP GET request, click **HTTP GET request**. By default, the HTTP GET request is targeted at the URL for the service endpoint followed by `?WSDL`. For example:

```
http://myhost:9080/WSSampleSei/EchoService?WSDL
```

   When the policy set attach point is at the application level you cannot change this value.
   a. Optional: If you are applying a policy to a service and the provider policy is located at the service endpoint, ensure that **Use the default request target** is selected.
   b. Optional: If you are applying a policy to a service and the provider policy is not located at the service endpoint, click **Specify request target**, then enter the URL for the location of the provider policy in the field. For example, you might change the target of the HTTP GET request if the provider policy is located in a repository.
   c. Optional: If you select **HTTP GET request** as the method to be used to obtain the provider policy and if you select **Specify request target** and you want to configure transport-level security, select **Attach a system policy set to the HTTP GET request**, then select a suitable policy set and binding from the drop-down lists. Select the policy set you require from the Policy set list to provide transport-level security for the HTTP GET request. Select from system policy sets that contain solely HTTP transport policies, solely SSL transport policies, or both; the policy set cannot contain

other policy types. Select the binding you require from the Binding list for the HTTP GET request. You can select from general bindings that are scoped to the global domain or scoped to the security domain of this service.

6. Optional: To obtain the provider policy by using a Web Services Metadata Exchange (WS-MetadataExchange) GetMetadata request, click **WS-MetadataExchange request**.
   a. Optional: If you select **WS-MetadataExchange request** and want to use message-level security, select **Attach a system policy set to the WS-MetadataExchange request**, then select a suitable policy set and binding from the drop-down lists. See “Configuring security for a WS-MetadataExchange request” on page 3115.
7. Click **OK**.
8. Save your changes to the master configuration.

Results

The web application client-side policy is calculated when it is required at run time, based either on the policy of the service provider, or on the client policy set and the policy of the service provider, depending on which option you selected. This calculated policy is known as the “effective policy” and is cached as a runtime configuration. The effective policy is used for subsequent outbound web service requests to the endpoint or operation for which the dynamic policy calculation was performed. The policy set configuration of the client does not change.

The provider policy that the client holds for a service is refreshed the first time that the web service is invoked after the application is loaded. After that, the provider policy is refreshed when the application restarts, or if the application explicitly invokes a refresh. When the provider policy is refreshed, the effective policy is recalculated.

Configuring the client policy to use a service provider policy by using wsadmin scripting

An application that is a web service client can obtain the policy configuration of a web service provider and use this information to establish a policy configuration that is acceptable to both the client and the service provider.

Before you begin

You have developed a web service client that contains all the necessary artifacts, and deployed your web services application into your application server instance. If you require them, you have attached the policy sets and managed the associated bindings.

The service provider must publish its policy in its Web Services Description Language (WSDL) and that policy must contain its policy configuration at run time in WS-PolicyAttachments format. The client must be able to support those provider policies.

For a list of WS-Policy assertion specifications and WS-Policy domains that are supported, see the WS-Policy topic.

About this task

You can administer the client to configure itself dynamically at run time, based on the policy of the service provider in the standard WS-PolicyAttachments format. You can administer the client to dynamically apply the provider policy at the application or service or service reference level.

Note: If you specify client dynamic policy control at the service reference level, you must use the new name-value paired list format of the resource string. If you are not specifying client dynamic policy control at service reference level, you can use either format.

Table 270. How to specify policy control at different levels of the application. For each applicable level of the application, the table lists the relevant string format command and name-value pair format command needed to specify policy control and summarizes the associated behavior.

Level	String format	Name-value pair list format (NEW)	Behavior
Type	"WebService:/"	"type=WebService:/"	Indicates all artifacts in the application
Service	"WebService:/*myModule*:{*namespace*}*myService*"	"type=WebService:/,module=*myModule*,service={*namespace*}*myService*"	Indicates all artifacts within the web service
Endpoint (under this service)	"WebService:/*myModule*:{*namespace*}*myService*/*endpointA*"	"type=WebService:/ ,module=*myModule*,service={*namespace*}*myService*,endpoint=*endpointA*"	Indicates all operations for this endpoint (under the service)
Operation (under this service)	"WebService:/*myModule*:{*namespace*}*myService*/*endpointA*/*operation1*"	"type=WebService:/ ,module=*myModule*,service={*namespace*}*myService*,endpoint=*endpointA*, operation=*operation1*"	Indicates a specific single operation (under the service)
Service reference	[Not possible]	"type=WebService:/ ,module=*myModule*,service={*namespace*}*myService*,serviceRef=*myServiceRef*"	Indicates all artifacts within the web service reference
Endpoint (under this service reference)	[Not possible]	"type=WebService:/ ,module=*myModule*,service={*namespace*}*myService*,serviceRef=*myServiceRef*, endpoint=*endpointA*"	Indicates all operations for this endpoint (under the service reference)
Operation (under this service reference)	[Not possible]	"type=WebService:/ ,module=*myModule*,service={*namespace*}*myService*,serviceRef=*myServiceRef*, endpoint=*endpointA*,operation=*operation1*"	Indicates a specific single operation (under the service reference)

If the provider policy uses multipart WSDL, you can use an HTTP GET request to obtain the policy of the provider, but you cannot use the WS-MetadataExchange protocol. For more information about multipart WSDL, see the topic about WSDL.

Policy intersection is the comparison of a client policy and a provider policy to determine whether they are compatible, and the calculation of a new policy, known as the effective policy, that complies with both their requirements and capabilities.

This topic describes how to configure the client policy to use a service provider policy by using wsadmin commands. You can also configure the client policy to use a service provider policy by using the administrative console.

Procedure

1. Start the wsadmin scripting client if it is not already running.
2. Use the SetClientDynamicPolicyControl command. For example:

```
AdminTask.setClientDynamicPolicyControl('[-applicationName WebServiceClientApplication
-resource WebService:/ClientApplication.war:{http://example_path/}Service1
-acquireProviderPolicyMethod [httpGet ]
-httpGetProperties [httpGetTargetURI http://example_path]]')
```

3. Save your changes to the master configuration.

   To save your configuration changes, enter the following command:

```
AdminConfig.save()
```

Results

The web application client-side policy is calculated when it is required at run time, based either on the policy of the service provider, or on the client policy set and the policy of the service provider, depending on which option you selected. This calculated policy is known as the “effective policy” and is cached as a runtime configuration. The effective policy is used for subsequent outbound web service requests to the endpoint or operation for which the dynamic policy calculation was performed. The policy set configuration of the client does not change.

The provider policy that the client holds for a service is refreshed the first time that the web service is invoked after the application is loaded. After that, the provider policy is refreshed when the application restarts, or if the application explicitly invokes a refresh. When the provider policy is refreshed, the effective policy is recalculated.

Configuring the client policy to use a service provider policy from a registry

An application that is a web service client can obtain the policy configuration of a web service provider from a registry, such as WebSphere Service Registry and Repository (WSRR), and use this information to establish a policy configuration that is acceptable to both the client and the service provider.

Before you begin

You have developed a web service client that contains all the necessary artifacts, and deployed your web services application into your application server instance. If you require them, you have attached the policy sets and managed the associated bindings.

The Web Services Description Language (WSDL) for the policy of the service provider, and its corresponding policies and policy attachments, are stored in a registry such as WSRR. That policy must contain its policy configuration in WS-PolicyAttachments format. The client must be able to support those provider policies.

The registry must support the use of HTTP GET requests to publish WSDL that contains WS-Policy attachments, for example WSRR Version 6.2 or later.

For a list of WS-Policy assertion specifications and WS-Policy domains that are supported, see the WS-Policy topic.

About this task

You can administer the client to configure itself dynamically at run time, based on the policy of a service provider that is held in a registry. You can administer the client at the service or service reference level to dynamically apply the provider policy that it obtains from a registry. By default, endpoints and operations inherit their policy configuration from the relevant service. However, it is possible to configure a service reference to override the service, in which case the endpoints and operations inherit their policy configuration from the service reference. You cannot administer the client to apply dynamically the provider policy that it obtains from a registry at the application level.

You can configure the client policy to use a service provider policy that is stored in a registry by using the administrative console. You can also configure the client policy to use a service provider policy that is stored in a registry by using wsadmin commands.

Procedure

1. From the navigation pane of the administrative console, click **Applications** > **Application Types** > **WebSphere enterprise applications**.
2. Click the web service client application that you want to configure.
3. Click **[Web services properties] Service client policy sets and bindings**.
4. In the row for the service where you want to apply the policy, click the link in the Policies Applied column. You cannot apply the policy at application level. The Policies Applied pane is displayed.
5. Select one of the following options from the drop-down list:
   - Provider policy only. Configure the client based solely on the policy of the service provider. This option is available when a client policy set is not attached.
   - Client and provider policy. Configure the client based on both the client policy set and the policy of the service provider. This option is available when a client policy set is attached.

   The other options in the list do not apply to this task.
6. Click **HTTP GET request**.
7. Click **Specify request target**, then enter the URL for the location of the provider policy in the field, that is, the address in the repository for the WSDL and policy. For information about using WSRR to

retrieve a WSDL document with embedded policies, and therefore obtain the required URL, see the WSRR documentation. The following example shows a typical URL:

```
https://www.wsrr.host/WSRR/6.2/PolicyService/
WSDL?bsrURI=3b9b493b-278f-4f64.ba3f.dabd30da3f7e
```

8. Click **OK**.
9. Optional: If there is a secure connection that uses the Secure Sockets Layer (SSL) protocol between the client and the registry, ensure that trust is established between the application server and the registry server. To access the registry, the client uses the SSL transport policy that is part of its service-level application policy. For example, for WSRR, you can enter the URL for the WSRR server in a browser window. If the WSRR server is not already trusted, a message is displayed stating that the security certificate is not trusted. To establish trust, use the following steps:
   a. Retrieve and store the X509 certificate from the WSRR server. Use the options on the message to view details of the certificate and save those details to a file, using distinguished encoding rules (DER) encoded binary format.
   b. Find out the key store that the client uses, that is, the key store that is shown by the SSL security transport bindings of the client application policy set. See Configuring the SSL transport policy. For example, the key store might be the default trust store for the node.
   c. Add the signer certificate that you saved in step a. to the key store that the client uses. See "Adding a signer certificate to a keystore" on page 1824.
10. Optional: To access the registry, the client uses the transport policy that is part of its service-level application policy. If the registry requires authentication using the HTTP protocol, configure a valid user name and password as part of the application-level transport policy binding configuration. It is advisable to secure any authorization credentials, because they are used for interactions with both the web service endpoint and the registry.
    a. Ensure that the client has a policy set that contains the HTTP transport policy attached to the application or service level. See the relevant steps in "Managing policy sets and bindings for service clients at the application level using the administrative console" on page 2656.
    b. Configure the HTTP transport client bindings for the binding named Client sample and enter the user name and password that the registry requires to authenticate outbound service requests. See the relevant steps in "Configuring the HTTP transport policy" on page 2720.
11. Save your changes to the master configuration.

Results

The web application client-side policy is calculated when it is required at run time, based either on the policy of the service provider, or on the client policy set and the policy of the service provider, depending on which option you selected. This calculated policy is known as the "effective policy" and is cached as a runtime configuration. The effective policy is used for subsequent outbound web service requests to the endpoint or operation for which the dynamic policy calculation was performed. The policy set configuration of the client does not change.

The provider policy that the client holds for a service is refreshed the first time that the web service is invoked after the application is loaded. After that, the provider policy is refreshed when the application restarts, or if the application explicitly invokes a refresh. When the provider policy is refreshed, the effective policy is recalculated.

Policies applied settings

Use this panel to view and change whether the policy configuration of a WebSphere Application Server service client is configured dynamically, based on the policies supported by its service provider. You can view or change how the client obtains the policy of the service provider; the client can use an HTTP GET request or a Web Services Metadata Exchange (WS-MetadataExchange) request. You can specify a policy set and binding to provide message-level security for WS-MetadataExchange requests or to specify HTTP transport and SSL transport configuration for HTTP GET requests.

To view this administrative console page for an application or service, complete the following steps. The application must be a web services service client.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***service_client_application_name*** > **[Web Services Properties] Service client policy sets and bindings**.
2. Select the link in the **Policies Applied** column for the application or service you require.

To view this administrative console page for a service, complete the following steps:

1. Click **Services** > **Service clients** > ***service_name***.
2. Select the link in the **Policies Applied** column for the service. The link is available if the service or the parent application has a policy set attached.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Apply the following policies:

Specifies whether the client policy is based on the policy of the service provider, and how that policy is used.

Select from the following options:

- Client policy only. Configure the client based solely on the client policy set. Do not use the policy of the service provider. This option is available when a client policy set is attached to the resource.
- Client and provider policy. Configure the client based on both the client policy set and the policy of the service provider. This option is available when a client policy set is attached to the resource.
- Inherit application attachment. Inherit the setting of the parent application. This option is available for a service when a client policy set is not attached to the service. If there is a policy set attached to the parent application, the inherited properties are displayed on this panel, but you cannot change them. If there is no policy set attached to the parent application, when you return to the Service clients policy sets and bindings panel, the Policies applied column shows a value of **None**.
- Provider policy only. Configure the client based solely on the policy of the service provider. This option is available when a client policy set is not attached to the resource.

HTTP GET request:

Click **HTTP GET request** to obtain the policy of the service provider by using an HTTP GET request. The policy configuration must be in WS-PolicyAttachments format in the WSDL of the service provider.

This option is available when **Apply the following policies** is set to Client and provider policy or Provider policy only.

By default, the HTTP GET request is targeted at the URL for each service endpoint followed by ?WSDL.

Use the default request target:

When you apply a policy to a service, click **Use the default request target** to target the HTTP GET request at the URL for each service endpoint followed by ?WSDL.

If the attach point is for the service, then you can either select this default request target or you can choose to specify an alternative request target using the **Specify request target** option.

If the attach point is for the application then the default request target will be used.

Specify request target:

When you apply a policy to a service, click **Specify request target** to change the target for acquiring provider policy using an HTTP GET request. Enter the URL for the location of the provider policy in the field.

This option is available when **HTTP GET request** is selected and you apply a policy to a service.

When you apply a policy to an application, this option is not available.

Attach a system policy set to the HTTP GET request:

Select **Attach a system policy set to the HTTP GET request** to set HTTP transport and SSL transport configuration for the HTTP GET request. This option is available when **HTTP GET request** is selected as the method to be used to obtain the provider policy and when **Specify request target** is selected and completed.

If you do not specify a policy set you will inherit the HTTP transport and SSL transport configuration from the application.

Policy set (for the HTTP GET request):

Select the policy set you require from the list to provide HTTP transport and SSL transport configuration for the HTTP GET request. Select from system policy sets that contain solely HTTP transport policies, solely SSL transport policies, or both; the policy set cannot contain other policy types.

This option is available when **Attach a system policy set to the HTTP GET request** is selected and the **Specify request target** is selected and completed.

Binding (for the HTTP GET request):

Select the binding you require from the list for the HTTP GET request. You can select from **Global Default Bindings** or **General client/provider policy set bindings**, which are specific to the individual service.

This option is available when **Attach a system policy set to the HTTP GET request** is selected and the **Specify request target** is selected and completed.

The value of **Default** will result in the Global Default Binding being used.

WS-MetadataExchange request:

Click **WS-MetadataExchange** to obtain the policy of the service provider by using a WS-MetadataExchange GetMetadata request. The policy configuration must be in WS-PolicyAttachments format in the WSDL of the service provider.

This option is available when **Apply the following policies** is set to Client and provider policy or Provider policy only.

Attach a system policy set to the WS-MetadataExchange request:

Select **Attach a system policy set to the WS-MetadataExchange request** to set message-level security for the WS-MetadataExchange request. By default, this option is not selected and the transport policy of the application is used. This option is available when **WS-MetadataExchange request** is selected.

When **Attach a system policy set to the WS-MetadataExchange request** is selected, the **Policy set** and **Binding** lists are available. If you select **Attach a system policy set to the WS-MetadataExchange request**, you must also select a policy set and a binding.

Policy set (for the WS-MetadataExchange request):

Select the policy set you require from the list to provide message-level security for the WS-MetadataExchange request. You can select from system policy sets that contain only WS-Security policies, only WS-Addressing policies, or both. The default policy set is SystemWSSecurityDefault.

System policy sets are used for system messages that are not business-related, for example, messages that apply qualities of service (QoS), including the messages that are defined in the WS-MetadataExchange protocol.

Note that any transport policy of the application is always used.

This option is available when **Attach a system policy set to the WS-MetadataExchange** is selected.

Binding (for the WS-MetadataExchange request):

Select the binding you require from the list to provide message level security for the WS-MetadataExchange request. You can select from **Global Default Bindings** or **General client/provider policy set bindings**, which are specific to the individual service.

This option is available when **Attach a system policy set to the WS-MetadataExchange** is selected.

The value of **Default** will result in the Global Default Binding being used.

Configuring security for a WS-MetadataExchange request

You can configure message-level security for a Web Services Metadata Exchange (WS-MetadataExchange) GetMetadata request by specifying a suitable policy set and binding. You do this when you configure a web service provider to share its policies or a web service client to obtain the policies of a service provider.

Before you begin

For a service provider, you have completed the procedure to configure a service provider to share its policy configuration, up to and including the step to enable WS-MetadataExchange.

For a service client, you have completed the procedure to configure the client policy to use a service provider policy, up to and including the step to use WS-MetadataExchange.

About this task

By default, the WS-MetadataExchange GetMetadata request uses the transport-level security configuration of the application. You might want to apply message-level security if transport-level security is not available on the application endpoint, or if transport-level security is not adequate for your requirements. An advantage of message-level security is that it provides end-to-end security, which is especially important for the exchange of security metadata.

You can configure security for a WS-MetadataExchange request by using the administrative console. You can also configure security for a WS-MetadataExchange request by using wsadmin commands.

Procedure

1. For a service provider, in the Policy Sharing panel on the administrative console, select **Attach a system policy set to the WS-MetadataExchange**. For a service client, in the Policies Applied panel on the administrative console, select **Attach a system policy set to the WS-MetadataExchange**.
2. Select a system policy set to provide message-level security from the Policy set list. You can select from system policy sets that contain only WS-Security policies, only WS-Addressing policies, or both. The default policy set is SystemWSSecurityDefault. If the policy sets that are listed are not suitable for your requirements, create your own system policy set, then return to this procedure.
3. Select a general binding for the policy set attachment from the Binding list. You can select from general bindings that are scoped to the global domain, or the security domain of this service. If the bindings that are listed are not suitable for your requirements, create your own general binding, then return to this procedure.
4. Click **OK**.
5. Save your changes to the master configuration.

Results

Message-level security is applied to the WS-MetadataExchange GetMetadata request.

Chapter 34. Administering web services - Reliable messaging (WS-ReliableMessaging)

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent.

Administering reliable web services

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent.

About this task

WS-ReliableMessaging is an interoperability standard for the reliable transmission of messages between two endpoints. To administer WS-ReliableMessaging for an application, you take the following broad actions:

1. Attach a reliable messaging policy set (either a default policy set or one that you have created) to an aspect of your application (that is, application level or web service level). Policy sets define the reliability level (quality of service) and other configuration options that you want to apply to your reliable messaging application.
2. Define the bindings for each attachment to a policy set that specifies a managed quality of service. That is, choose the service integration bus and messaging engine to use to maintain the state for the managed persistent and managed non-persistent qualities of service.

At any stage, you can set a property that configures endpoints to only support clients that use reliable messaging. This setting is reflected by WS-Policy if engaged.

If your WS-ReliableMessaging application runs inside the web container and uses a managed quality of service, you can also use WS-ReliableMessaging to provide transactional recoverable messaging.

Procedure

- Configure a WS-ReliableMessaging policy set.

  You can do this using the administrative console or using the wsadmin tool. You can also configure WS-SecureConversation to work with WS-ReliableMessaging.
- Attach and bind a WS-ReliableMessaging policy set to a web service application

  You can do this using the administrative console or using the wsadmin tool.
- Configure endpoints to only support clients that use reliable messaging.
- Provide transactional recoverable messaging through WS-ReliableMessaging.

Configuring a WS-ReliableMessaging policy set by using the administrative console

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent. Use the administrative console to configure a policy set for reliable messaging.

Before you begin

You can configure a reliable messaging policy set by using the administrative console as described in this task, or you can configure a reliable messaging policy set by using the wsadmin tool.

The following default policy sets work with WS-ReliableMessaging applications:

- WS-I RSP
- WS-I RSP ND
- LTPA WS-I RSP
- Username WS-I RSP
- WSReliableMessaging 1_0
- WSReliableMessaging default
- WSReliableMessaging persistent

For more information, see “WS-ReliableMessaging default policy sets” on page 2671.

If you can use any of these default policy sets without needing to modify their configuration, you need not complete this task. You are ready to attach and bind the default policy set to your application.

At any stage - that is, before or after you have built your reliable web service application, or configured your policy sets - you can set a property that configures endpoints to only support clients that use reliable messaging. This setting is reflected by WS-Policy if engaged.

About this task

To configure a reliable messaging policy set by using the administrative console, complete the following steps:

Procedure

1. Create a policy set. You can create a new policy set, or copy and rename an existing policy set - either one that you have previously created, or one of the WS-ReliableMessaging default policy sets.
2. Check that your policy set includes the policy types **WS-ReliableMessaging** and **WS-Addressing**. Add these policy types if necessary. These policy types contain the configuration options that support WS-ReliableMessaging. WS-Addressing provides the asynchronous request and reply capabilities for WS-ReliableMessaging, and is also required for WS-ReliableMessaging Version 1.1 synchronous messaging.

   Notes:

   - If you want to use secure conversation and reliable messaging policies in the same policy set, the secure conversation bindings must be configured to require that the reliable messaging headers are signed. The reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND) are specifically designed and configured to use secure conversation and reliable messaging in the same policy set. If you use a copy of one of the reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND), no further configuration of the secure conversation bindings is required. Otherwise, see “Configuring WS-SecureConversation to work with WS-ReliableMessaging” on page 3121.
   - WS-ReliableMessaging Version 1.1 messaging requires WS-Addressing to be mandatory. If you use a policy set that includes WS-ReliableMessaging and WS-Addressing policies, and the WS-Addressing policy is configured as optional, then WebSphere Application Server overrides the WS-Addressing setting and automatically enables WS-Addressing.
3. Configure the **WS-ReliableMessaging** policy type attributes. For the WS-ReliableMessaging policy you can configure the version of the WS-ReliableMessaging standard that you want to use, the order in which messages are delivered, and the required quality of service (the reliability level) for message delivery.

   Note: In WebSphere Application Server Version 6.1, you could also configure whether or not to use the WS-MakeConnection protocol. This configuration option has now been removed from the

administrative console panel, because the product now automatically determines whether WS-MakeConnection is used, based on the following criteria:

- Whether you are using WS-ReliableMessaging Version 1.0 or Version 1.1.
- Whether the requester supports WS-MakeConnection.
- Whether the message exchange protocol is synchronous or asynchronous.

4. If required, configure the **WS-Addressing** policy type attributes. For example, the default WS-Addressing policy messaging style is **Synchronous and asynchronous**, which specifies that there is no restriction on the targeting of response messages. However if you enable policy sharing, the WS-Policy framework determines which style to use, and has a preference for the synchronous request-response pattern. Because the WS-Policy framework takes precedence, reliable messages are sent in a synchronous request-response pattern even if your client invokes the service asynchronously. To enforce asynchronous messaging, set the WS-Addressing policy messaging style to asynchronous only.
5. Save your changes to the master configuration.

What to do next

You are now ready to attach your application to the policy set and define the bindings that you want to use.

Configuring a WS-ReliableMessaging policy set by using the wsadmin tool

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent. Use command scripts to configure a policy set for reliable messaging.

Before you begin

You can configure a reliable messaging policy set by using the wsadmin tool as described in this task, or you can configure a reliable messaging policy set by using the administrative console.

The following default policy sets work with WS-ReliableMessaging applications:

- WS-I RSP
- WS-I RSP ND
- LTPA WS-I RSP
- Username WS-I RSP
- WSReliableMessaging 1_0
- WSReliableMessaging default
- WSReliableMessaging persistent

For more information, see “WS-ReliableMessaging default policy sets” on page 2671.

If you can use any of these default policy sets without needing to modify their configuration, you need not complete this task. You are ready to attach your application to the default policy set and define the bindings that you want to use.

At any stage - that is, before or after you have built your reliable web service application, or configured your policy sets - you can set a property that configures endpoints to only support clients that use reliable messaging. This setting is reflected by WS-Policy if engaged.

About this task

To configure a reliable messaging policy set by using the wsadmin tool, complete the following steps:

Procedure

1. Create a policy set. Use the createPolicySet command to create a new policy set, or the copyPolicySet command to copy and rename an existing policy set - either one that you have previously created, or one of the two “WS-ReliableMessaging default policy sets” on page 2671. For more information, see Creating and copying policy sets by using the wsadmin tool.
2. If the policy set does not include both the policy types WSReliableMessaging and WSAddressing, add these policy types by using the addPolicyType command as described in Creating and copying policy sets by using the wsadmin tool. For example:

```
AdminTask.addPolicyType('[-policySet PolicySet1 -policyType WSReliableMessaging]')
AdminTask.addPolicyType('[-policySet PolicySet1 -policyType WSAddressing]')
```

   These policy types contain the configuration options that support WS-ReliableMessaging. WS-Addressing provides the asynchronous request and reply capabilities for WS-ReliableMessaging, and is also required for WS-ReliableMessaging Version 1.1 synchronous messaging.

   Notes:

   - If you want to use secure conversation and reliable messaging policies in the same policy set, the secure conversation bindings must be configured to require that the reliable messaging headers are signed. The reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND) are specifically designed and configured to use secure conversation and reliable messaging in the same policy set. If you use a copy of one of the reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND), no further configuration of the secure conversation bindings is required. Otherwise, see “Configuring WS-SecureConversation to work with WS-ReliableMessaging” on page 3121.
   - WS-ReliableMessaging Version 1.1 messaging requires WS-Addressing to be mandatory. If you use a policy set that includes WS-ReliableMessaging and WS-Addressing policies, and the WS-Addressing policy is configured as optional, then WebSphere Application Server overrides the WS-Addressing setting and automatically enables WS-Addressing.

3. Configure the **WS-ReliableMessaging** policy type attributes.

   For the WS-ReliableMessaging policy you can configure the version of the WS-ReliableMessaging standard that you want to use, the order in which messages are delivered, and the required quality of service (the reliability level) for message delivery. For detailed information about these configurable attributes, see “WS-ReliableMessaging settings” on page 2716.

   Use the setPolicyType command to configure these attributes. For example:

```
AdminTask.setPolicyType('-policySet PolicySet1 -policyType WSReliableMessaging -attributes "[[inOrderDelivery false][specLevel 1.0]
[enabled true][qualityOfService managedPersistent][type WSReliableMessaging]]" -replace'
```

4. If required, configure the **WS-Addressing** policy type attributes. For example, the default WS-Addressing policy messaging style is **Synchronous and asynchronous**, which specifies that there is no restriction on the targeting of response messages. However if you enable policy sharing, the WS-Policy framework determines which style to use, and has a preference for the synchronous request-response pattern. Because the WS-Policy framework takes precedence, reliable messages are sent in a synchronous request-response pattern even if your client invokes the service asynchronously. To enforce asynchronous messaging, set the WS-Addressing policy messaging style to asynchronous only.

   Use the setPolicyType command to configure these attributes. For example:

```
AdminTask.setPolicyType('[-policySet PolicySet1 -policyType WSAddressing
-attributes "[[wsaMode WSA_ASYNC]]"]')
```

   For detailed information about these configurable attributes, see “WS-Addressing policy settings” on page 2719.

5. Save your changes to the master configuration.

   To save your configuration changes, enter the following command:

```
AdminConfig.save()
```

What to do next

You are now ready to attach your application to the default policy set and define the bindings that you want to use.

Configuring WS-SecureConversation to work with WS-ReliableMessaging

Configure secure conversation to expect the reliable messaging headers to be signed, and to ensure that the scoping security context token does not expire before reliable messaging recovers and resends persistent messages.

Procedure

- "Configure secure conversation to expect the reliable messaging headers to be signed"
- "Configure secure conversation to increase the timeout setting for the scoping security context token"

Configure secure conversation to expect the reliable messaging headers to be signed:
About this task

Although secure conversation allows message headers to remain unsigned, the reliable messaging policy requires the reliable messaging headers to be signed. If you want to use secure conversation and reliable messaging policies in the same policy set, the secure conversation bindings must be configured to require that the reliable messaging headers are signed. To achieve this, complete one of the following steps:

Procedure

- Create your policy set from a copy of one of the reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND). These policy sets contain instances of the secure conversation and reliable messaging policies that are configured to work together.
- Create your policy set from a copy of the secure conversation policy set:
  1. Add the reliable messaging policy to your copy of the secure conversation policy set.
  2. Specify in the secure conversation bindings that the reliable messaging headers must be signed. For more information about how to do this, see "Defining and managing policy set bindings" on page 2680.
  3. Save your changes to the master configuration.

  Note: The reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND) are specifically designed and configured to use secure conversation and reliable messaging in the same policy set. Only create your policy set from a copy of the secure conversation policy set if you have a clear business need to do so. Otherwise, always use a copy of one of the reliable secure profile default policy sets (WS-I RSP and WS-I RSP ND).

Configure secure conversation to increase the timeout setting for the scoping security context token:
About this task

When you use a persistent WS-I RSP policy set, which includes WS-SecureConversation, if the scoping security context token is expired when the server is restarted then WS-ReliableMessaging cannot resend its messages and system messages are written to the log file stating that the reliable messaging sequence was not secured using the correct security token.

To ensure that the scoping security context token does not expire before WS-ReliableMessaging can recover and resend its messages, use the administrative console to complete the following steps:

Procedure

1. In the navigation pane, click **Services** > **Security cache**. The Security cache detail form is displayed.

2. Set the **Time token is in cache after timeout** property to a value of at least 120 minutes. This value specifies the length of time to keep tokens after they expire. By default, this is 10 minutes. These expired tokens can be used for reliable messaging recovery.
3. In the navigation pane, click **Services** > **Trust service** > **Token providers**. In the content pane, select a security context token. The Security Context Token detail form is displayed.
4. Set the following values for the security context token:
   a. Check that the **Time in cache after expiration** property is set to a value of at least 120 minutes.
   b. Check that the **Token timeout** property is set to a value of at least 120 minutes.
   c. Select the **Allow renewal after timeout** check box.
5. Save your changes to the master configuration.

Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the administrative console

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent. Use the administrative console to attach the policy set to your application, and (for managed qualities of service) define bindings to a service integration bus and messaging engine.

Before you begin

You can attach a WS-ReliableMessaging policy set and define bindings by using the administrative console as described in this task, or you can attach and bind a WS-ReliableMessaging policy set by using the wsadmin tool.

This task assumes that you have already developed and installed the web service application to which you want to attach a policy set.

The following default policy sets work with WS-ReliableMessaging applications:

- WS-I RSP
- WS-I RSP ND
- LTPA WS-I RSP
- Username WS-I RSP
- WSReliableMessaging 1_0
- WSReliableMessaging default
- WSReliableMessaging persistent

For more information, see “WS-ReliableMessaging default policy sets” on page 2671.

If you can use any of these default policy sets, or you have configured your own reliable messaging policy set, then you are ready to complete this task.

About this task

Use this task to complete the following broad actions:

1. Attach a reliable messaging policy set (either a default policy set or one that you have created) to an aspect of your application (that is, application level or web service level). Policy sets define the reliability level (quality of service) and other configuration options that you want to apply to your reliable messaging application.
2. Define the bindings for each attachment to a policy set that specifies a managed quality of service. That is, choose the service integration bus and messaging engine to use to maintain the state for the managed persistent and managed non-persistent qualities of service.

To attach a WS-ReliableMessaging policy set and define bindings by using the administrative console, complete the following steps:

Procedure

1. Attach a policy set to your reliable messaging application at either application level or service level..

   Note:

   - You can attach one policy set at each level.
   - You can only apply a WS-ReliableMessaging policy at application level or service level.
   - If you apply reliable messaging at service level, then all services must use the same WS-ReliableMessaging policy and bindings values.
   - You can attach any policy set at operation level. For a policy set that includes the WS-ReliableMessaging policy, attachment at the operation level configures the other components of the policy set (for example WS-Security and WS-Addressing) but any WS-ReliableMessaging configuration at operation level is ignored.

2. If your chosen policy set specifies a managed quality of service, define bindings to a service integration bus and messaging engine.

   If the policy set instance specifies managed non-persistent or managed persistent quality of service, choose the service integration bus and messaging engine that is to manage the WS-ReliableMessaging state. Use the "WS-ReliableMessaging policy binding" on page 2718 panel to select or create the service integration bus and messaging engine that you want to use.

   Note: When many applications use the same messaging engine, it can impact performance. Factors to consider include the number of applications that are already binding to the messaging engine, the CPU utilization, and the message throughput. To improve performance for a single server configuration, create a new messaging engine to bind to your application.

   To define the default WS-ReliableMessaging policy binding for provider and client policy set attachments within WebSphere Application Server Version 6.1 applications, and for attachments to service applications that are deployed to a Version 6.1 server, navigate to **Services** > **Policy sets** > **Default policy set bindings** > **Version 6.1 default policy set bindings** > **WS-ReliableMessaging**.

   To define the bindings for a WebSphere Application Server Version 7.0 or later provider or client policy set, navigate to **Services** > **Policy sets** > **General provider policy set bindings** > ***provider_policy_set_binding_name*** > **WS-ReliableMessaging** or **Services** > **Policy sets** > **General client policy set bindings** > ***client_policy_set_binding_name*** > **WS-ReliableMessaging**

   To define the bindings for an application that you have attached to a service provider policy set, navigate to **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **[Web Service Properties] Service provider policy sets and bindings** and follow the instructions given in "Managing policy sets and bindings for service providers at the application level using the administrative console" on page 2634.

   To define the bindings for an application that you have attached to a service client policy set, navigate to **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** > **[Web Service Properties] Service client policy sets and bindings** and follow the instructions given in "Managing policy sets and bindings for service clients at the application level using the administrative console" on page 2656.

   WS-Notification note: If the application that you have attached to a service client policy set is a Version 7.0 WS-Notification service client, you can instead use the context-specific version of the "Service client policy sets and bindings" panel that can be reached through either of the following paths:

   - **Service integration** > **WS-Notification** > **Services** > ***service_name*** > **[Additional properties] Outbound request policy sets and bindings**
   - **Service integration** > **Buses** > ***bus_name*** > **[Services] WS-Notification services** > ***service_name*** > **[Additional properties] Outbound request policy sets and bindings**

If you want to configure policy set and binding details for a single Version 7.0 WS-Notification service client, rather than for all clients for the service, you can instead use the following panel:

- **Services** > **Service clients** > ***ws-notification_service_client_name***

This panel also gives you links to the associated service integration bus and WS-Notification service.

3. Save your changes to the master configuration.

Attaching and binding a WS-ReliableMessaging policy set to a web service application by using the wsadmin tool

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent. Use the wsadmin tool to attach the policy set to your application, and (for managed qualities of service) to define bindings to a service integration bus and messaging engine.

Before you begin

You can attach a WS-ReliableMessaging policy set and define bindings by using the wsadmin tool as described in this task, or you can attach and bind a WS-ReliableMessaging policy set by using the administrative console.

This task assumes that you have already developed and installed the web service application to which you want to attach a policy set.

The following default policy sets work with WS-ReliableMessaging applications:

- WS-I RSP
- WS-I RSP ND
- LTPA WS-I RSP
- Username WS-I RSP
- WSReliableMessaging 1_0
- WSReliableMessaging default
- WSReliableMessaging persistent

For more information, see “WS-ReliableMessaging default policy sets” on page 2671.

If you can use any of these default policy sets, or you have configured your own reliable messaging policy set, then you are ready to complete this task.

About this task

Use this task to complete the following broad actions:

1. Attach a reliable messaging policy set (either a default policy set or one that you have created) to an aspect of your application (that is, application level or web service level). Policy sets define the reliability level (quality of service) and other configuration options that you want to apply to your reliable messaging application.
2. Define the bindings for each attachment to a policy set that specifies a managed quality of service. That is, choose the service integration bus and messaging engine to use to maintain the state for the managed persistent and managed non-persistent qualities of service.

To attach a WS-ReliableMessaging policy set and define bindings by using the wsadmin tool, complete the following steps:

Procedure

1. Attach a policy set to your reliable messaging application at either application level or service level.

Use the createPolicySetAttachment command as described in Creating policy set attachments by using the wsadmin tool. Set the **-policySet** parameter to the name of the reliable messaging policy set that you want to use. For example: WS-I RSP ND

Note:

- You can attach one policy set at each level.
- You can only apply a WS-ReliableMessaging policy at application level or service level.
- If you apply reliable messaging at service level, then all services must use the same WS-ReliableMessaging policy and bindings values.
- You can attach any policy set at operation level. For a policy set that includes the WS-ReliableMessaging policy, attachment at the operation level configures the other components of the policy set (for example WS-Security and WS-Addressing) but any WS-ReliableMessaging configuration at operation level is ignored.

This command returns an attachment ID number. If your chosen policy set specifies a managed quality of service, make a note of this number. In the next step you use it to define the binding.

2. If your chosen policy set specifies a managed quality of service, define bindings to a service integration bus and messaging engine.

   If the policy set instance specifies managed non-persistent or managed persistent quality of service, choose the service integration bus and messaging engine that is to manage the WS-ReliableMessaging state. Use the setBinding command as described in Creating policy set attachments using the wsadmin tool. Set the **-policyType** parameter to WSReliableMessaging. Set the bus name and the messaging engine name by using the following syntax for the **-attributes** parameter:

```
-attributes "[[busName ReliableMessagingBus]
             [messagingEngineName messaging_engine_name]]"
```

   The messaging engine name is in the format *nodeName.serverName-busName* for a messaging engine on a single server, or *clusterName.nnn-busName* for a messaging engine in a cluster.

   Note: When many applications use the same messaging engine, it can impact performance. Factors to consider include the number of applications that are already binding to the messaging engine, the CPU utilization, and the message throughput. To improve performance for a single server configuration, create a new messaging engine to bind to your application.

3. Save your changes to the master configuration.

   To save your configuration changes, enter the following command:

```
AdminConfig.save()
```

Configuring endpoints to only support clients that use WS-ReliableMessaging

By default, when a WS-ReliableMessaging enabled policy set is attached to an endpoint, the server supports clients that use reliable messaging and clients that do not use reliable messaging. In this version of the product, you can configure endpoints to only support clients that use reliable messaging.

About this task

You configure endpoints to only support clients that use reliable messaging by setting a property in either of the following ways:

- Set a property when packaging the application.
- Set a property as a JVM argument for the server.

This setting is reflected by WS-Policy if engaged. For information about how to engage WS-Policy, see “Using WS-Policy to exchange policies in a standard format” on page 3099.

Procedure

- When packaging the application, configure endpoints to only support clients that use reliable messaging by setting the **strictlyEnforceWSRM** property in the META-INF/MANIFEST.MF of a WAR file or EJB module.
- Using a JVM argument for the server, configure endpoints to only support clients that use reliable messaging by defining the Java virtual machine custom property **com.ibm.ws.websvcs.rm.strictlyEnforceWSRM** on the server. For more information, see Configuring the JVM.

Providing transactional recoverable messaging through WS-ReliableMessaging

If your WS-ReliableMessaging application runs inside the web container and uses a managed quality of service, you can use WS-ReliableMessaging to provide transactional recoverable messaging.

About this task

The WS-ReliableMessaging transactional model is as follows:

- On the web service requester side, the transaction is between the application and the local managed store.
- The WS-ReliableMessaging protocol delivers the message to the web service provider side, where a different transaction is used between the second managed store and the application being dispatched.

For the outbound (requestor) case on a one-way message send, if the **enableTransactionalOneWay** property is set to true, then the send is performed under any transactional context currently held by the application thread. (Note that transactions are not supported under an outbound two-way message exchange).

For the inbound (provider) case, if the **inOrderDelivery** property is set to true, then an inbound message is dispatched to the application under a transaction. For an inbound two-way message exchange, the response is also generated under that transaction and is not sent until that transaction has committed.

Note:

WS-ReliableMessaging transactions do not use the WS-AtomicTransactions protocol. The relationship between these two protocols is as follows:

- WS-AtomicTransactions and WS-ReliableMessaging are mutually exclusive when WS-ReliableMessaging is being used, with a managed store, to provide transactional recoverable messaging.
- If WS-ReliableMessaging is configured to use an in-memory store, then there are cases where a WS-AtomicTransaction can be flowed between the reliable messaging source and the reliable messaging destination for two-way invocations. In this situation, WS-ReliableMessaging only protects against network failures, not against server failure.

For more information, see WS-AtomicTransactions.

To provide transactional recoverable messaging through WS-ReliableMessaging, work through the steps described in Adding assured delivery to web services through WS-ReliableMessaging and also complete the following additional steps:

Procedure

- To enable transactional messaging for outbound (requester) one-way message sends, when you develop your JAX-WS web service application set the **enableTransactionalOneWay** property to Boolean.TRUE (or the string true) in the jaxWS request context map.

- To enable transactional messaging for inbound (provider) one-way and two-way message exchanges, when you configure your WS-ReliableMessaging policy either use the administrative console to select the option **Deliver messages in the order that they were sent** or use the wsadmin tool to set the **inOrderDelivery** property to true.

WS-ReliableMessaging - administrative console panels

Links to topics that describe the contents of the administrative console panels that you can use to configure and operate WS-ReliableMessaging. Each topic gives details of the purpose and use of a panel, the administrative console navigation path to the panel, and the values that you can set in each field of the panel.

Configuration panels

- "WS-ReliableMessaging settings" on page 2716
- "WS-ReliableMessaging policy binding" on page 2718

Runtime panels

This set of panels is available at any of the following scopes within the administrative console:

Cell: **Services** > **Reliable messaging state**

Application server:
Servers > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state**

Enterprise application:
Applications > **Enterprise Applications** > ***application_name*** > **[Web Services Properties] Reliable messaging state**

Bus: **Service integration** > **Buses** > ***bus_name*** > **[Services] Reliable messaging state**

Messaging engine:
Service integration > **Buses** > ***bus_name*** > **[Topology] Messaging engines** > ***messaging_engine_name*** > **[Additional Properties] Reliable messaging state**

At each of the previous scopes, the navigation structure is as follows:

- "Reliable messaging state settings" on page 3130
  - "Message store collection" on page 3130
  - "Inbound sequence collection" on page 3136
    - "Inbound sequences settings" on page 3138
      - "Acknowledgement state collection" on page 3140
      - "Inbound message collection" on page 3139
        - "Message settings" on page 3135
    - "Export messages settings" on page 3140
  - "Outbound sequence collection" on page 3131
    - "Outbound sequences settings" on page 3133
      - "Outbound message collection" on page 3135
        - "Message settings" on page 3135
    - "Export messages settings" on page 3140

WS-ReliableMessaging settings

For the WS-ReliableMessaging policy you can configure the version of the WS-ReliableMessaging standard that you want to use, the order in which messages are delivered, and the required quality of service (the reliability level) for message delivery. The product can enforce these policies on inbound messages and applies them to outbound messages.

To view this page in the console, click the following path: **Services** > **Policy sets** > **Application policy sets** > ***policy_set_name*** > **WS-ReliableMessaging**.

With WebSphere Application Server, you can use WS-ReliableMessaging with Java API for XML-Based Web Services (JAX-WS) web services applications that use a SOAP over HTTP binding. Select the WS-ReliableMessaging specification to use for reliable transmission of your messages.
WS-ReliableMessaging Version 1.1 is the default value. Select the WS-ReliableMessaging specification to use for reliable transmission of your messages. WS-ReliableMessaging Version 1.1 is the default value. Details of the supported WS-ReliableMessaging specifications are available at the following web addresses:

- The WS-ReliableMessaging specification Version 1.0, February 2005.
- The OASIS WS-ReliableMessaging specification Version 1.1, February 2007.

Note: If you plan to invoke a .NET-based web service, you must select WS-ReliableMessaging Version 1.0.

Do not edit the policies associated with the provided default policy sets. If you have to modify the reliable messaging policy settings, use a copy of a default policy set or create a new policy set.

At any stage - that is, before or after you have built your reliable web service application, or configured your policy sets - you can set a property that configures endpoints to only support clients that use reliable messaging. This setting is reflected by WS-Policy if engaged.

Standard:

Deliver messages in the order that they were sent:

Select this option if the sender of a request has to receive a response before it sends the next request.

If you enable in-order delivery, you must also ensure that the requester application polls for the messages in the order in which it is to receive them. For more information, see “Configuring the WS-ReliableMessaging policy” on page 2714.

Specifying in-order delivery also marginally increases reliability if you are using the managed persistent quality of service.

Quality of Service: Select one of the following qualities of service:

Unmanaged non-persistent - Tolerates network and remote system failures
You can configure web service applications to use WS-ReliableMessaging with a default in-memory store. This quality of service requires minimal configuration. However it is non-transactional and, although it allows for the resending of messages that are lost in the network, if a server becomes unavailable you will lose messages. The default is Unmanaged Non-Persistent.

Managed non-persistent - Tolerates system, network, and remote system failures, but state is discarded after messaging engine restart
This in-memory quality of service option uses a messaging engine to manage the sequence state, and messages are written to disk if memory is low. This quality of service allows for the re-sending of messages that are lost in the network, and can also recover from server failure. However, state is discarded after a messaging engine restart so in this case you will lose messages.

Managed persistent - Tolerates system, network, and remote system failures
This quality of service for asynchronous web service invocations is recoverable. This option also uses a messaging engine and message store to manage the sequence state. Messages are persisted at the web service requester server and at the web service provider server, and are recoverable if the server becomes unavailable. Messages that have not been successfully transmitted when a server becomes unavailable can continue to be transmitted after the server restarts.

Note:

- All three qualities of service are supported when applications are deployed to the application server. Thin client and client container applications use the first option only.
- For the unmanaged non-persistent quality of service, the messages are stored only in memory. For both of the managed qualities of service, the messages are managed by a messaging engine and stored in a message store. You specify a binding to a bus and messaging engine on the "WS-ReliableMessaging policy binding" on page 2718 form. If your chosen quality of service is Unmanaged Non-Persistent, which does not use a binding to a messaging engine, then any binding that you specify is ignored.

WS-ReliableMessaging policy binding

To configure a web service application to use WS-ReliableMessaging, you attach a policy set that contains a WS-ReliableMessaging policy type. This policy type offers a range of qualities of service: managed persistent, managed non-persistent, or unmanaged non-persistent. The managed qualities of service, managed persistent and managed non-persistent, are supported by the service integration bus. Use this page to select the bus and messaging engine to use for the reliable messaging protocol state.

To view this page in the console, click one of the following paths:

- **Services** > **Policy sets** > **Default policy set bindings** > **Version 6.1 default policy set bindings** > **WS-ReliableMessaging** (This is the default binding for client and provider policy set attachments within WebSphere Application Server Version 6.1 applications, and attachments to service applications that are deployed to a Version 6.1 server.)
- **Services** > **Policy sets** > **General provider policy set bindings** > ***provider_policy_set_binding_name*** > **WS-ReliableMessaging** (This binding is used for the specified provider policy set.)
- **Services** > **Policy sets** > **General client policy set bindings** > ***client_policy_set_binding_name*** > **WS-ReliableMessaging** (This binding is used for the specified client policy set.)

Note:

- You only have to specify a binding to a bus and messaging engine if you are using a managed quality of service. If your chosen quality of service is Unmanaged Non-Persistent, any binding that you specify is ignored. The quality of service is defined on the "WS-ReliableMessaging settings" on page 2716 form for your chosen policy set. For more information, see "Configuring the WS-ReliableMessaging policy" on page 2714.
- When many applications use the same messaging engine, it can impact performance. Factors to consider include the number of applications that are already binding to the messaging engine, the CPU utilization, and the message throughput. To improve performance for a single server configuration, create a new messaging engine to bind to your application.

Bus name:

Specifies a list of available service integration buses in the cell. Use the list to select a bus, or click **Manage buses, bus members, and messaging engines** to add a new bus. The bus that you add is selected for this binding configuration when you return to this panel.

Messaging engine:

Specifies a list of each bus member for the selected bus. Use the list to select a bus member, or click **Manage buses, bus members, and messaging engines** to add a new bus member. The bus member that you add is selected for this binding configuration when you return to this panel.

Reliable messaging state settings

This page provides an overview of the WS-ReliableMessaging runtime state. Use this page to manage reliable messaging at run time.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state**.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

Runtime tab: Runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

Reliable messaging state

Select the aspect of WS-ReliableMessaging for which you want to view runtime information.

The following icons are displayed here and on several other reliable messaging runtime panels:

	**OK**	Everything here, and (if there is a link) in all runtime panels below this link, is running normally.
	**Warning**	Something here, or (if there is a link) in one of the runtime panels below this link, is in an unusual state and you might have to take some action to resolve it. For example, the system might be awaiting a response from an endpoint. In this case, either the response will be received (in which case you need take no action and the runtime information will be updated to "OK") or the reliable messaging destination has stopped acknowledging messages (in which case you have to take some action to resolve the failed sequence).
	**Error**	There is a definite error that you must take some action to resolve, either here or (if there is a link) in one of the runtime panels below this link.

Note that for troubleshooting purposes you only have to follow links to the sub-panels if states other than "OK" are displayed.

Message stores
This page displays the collection of reliable messaging storage managers for the current scope.

Inbound sequences
This page displays the collection of inbound sequences for the current scope. Each inbound sequence is used to receive messages that have been transmitted reliably.

Outbound sequences
This page displays the collection of outbound sequences for the current scope. Outbound sequences are used to transmit messages reliably from the local application to the remote endpoint. Each sequence has a unique identifier.

Message store collection

This page displays the collection of reliable messaging storage managers for the current scope.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Message store**.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

The following icons are displayed here and on several other reliable messaging runtime panels:

	**OK**	Everything here, and (if there is a link) in all runtime panels below this link, is running normally.
	**Warning**	Something here, or (if there is a link) in one of the runtime panels below this link, is in an unusual state and you might have to take some action to resolve it. For example, the system might be awaiting a response from an endpoint. In this case, either the response will be received (in which case you need take no action and the runtime information will be updated to "OK") or the reliable messaging destination has stopped acknowledging messages (in which case you have to take some action to resolve the failed sequence).
	**Error**	There is a definite error that you must take some action to resolve, either here or (if there is a link) in one of the runtime panels below this link.

Note that for troubleshooting purposes you only have to follow links to the sub-panels if states other than "OK" are displayed.

Message store type
For the managed qualities of service, the messages are written to a messaging engine. For the unmanaged non-persistent quality of service, the messages are stored in memory.

Description
The quality of service being used and the application name.

Details
For in-memory stores the only possible value is "Running". For messages stored by a messaging engine, the possible values are "Running" or "Messaging engine not contactable", probably because the messaging engine is not running.

Status
The "OK" icon indicates that the message store is running. If the messaging engine is not contactable, the "Error" icon is displayed.

Outbound sequence collection

This page displays the collection of outbound sequences for the current scope. Outbound sequences are used to transmit messages reliably from the local application to the remote endpoint. Each sequence has a unique identifier.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Outbound sequences**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

The following icons are displayed here and on several other reliable messaging runtime panels:

	**OK**	Everything here, and (if there is a link) in all runtime panels below this link, is running normally.
	**Warning**	Something here, or (if there is a link) in one of the runtime panels below this link, is in an unusual state and you might have to take some action to resolve it. For example, the system might be awaiting a response from an endpoint. In this case, either the response will be received (in which case you need take no action and the runtime information will be updated to "OK") or the reliable messaging destination has stopped acknowledging messages (in which case you have to take some action to resolve the failed sequence).
	**Error**	There is a definite error that you must take some action to resolve, either here or (if there is a link) in one of the runtime panels below this link.

Note that for troubleshooting purposes you only have to follow links to the sub-panels if states other than "OK" are displayed.

Note:

You might see more sequences than you expect, due to sequence reallocation. If a sequence is reallocated, the original and new sequences are both visible.

Sequence identifier
This URI is the unique identifier for the sequence.

Associated application
The application that created the sequence.

Target endpoint URI
The destination to which messages are transmitted.

Message depth
The count of messages, held by the reliable messaging layer, that have not yet been transferred.

Messages sent
The total count of messages sent by the application for this sequence.

Details
The current state of the sequence. "Establishing" indicates that the sequence is awaiting a CreateSequenceResponse message from the reliable messaging destination; "Active" indicates that the sequence has been established; "Cannot contact the remote endpoint" indicates that the sequence has been established but the reliable messaging destination has stopped acknowledging messages; "Sequence closing" indicates that the sequence is closing and will accept no new messages; "Sequence closed" indicates that the sequence has been closed and will accept no new messages; "Sequence terminating" indicates that the sequence is terminating and will accept no new messages; "The inbound side has lost state in a terminate" indicates that the sequence has terminated with unacknowledged messages outstanding; "The sequence has timed out" indicates that the sequence has timed out.

Status
The icon indicates "OK", "Warning" or "Error" as previously described on this page. A more precise indication of the sequence state is given in the "Details" column.

Buttons

Export unsent messages	Export the messages from the selected sequences to compressed files. If you select any sequences that have no messages to export, a warning or error message is displayed.

Close sequence	Close the selected sequences and send a CloseSequence message to indicate that the sequence is complete and will not be sending any further messages. The *close* protocol does not delete the resources for the sequence, so you can still access any undispatched or unsent messages. If you are using Version 1.0 of the WS-ReliableMessaging specification (which does not support this option) an error message is displayed. For more information about the reliable messaging *close* protocol, see WS-ReliableMessaging: supported specifications and standards.
Terminate sequence	Close the selected sequences and send a TerminateSequence message to indicate that the sequence is complete and will not be sending any further messages. The *terminate* protocol deletes all resources for the sequence. For more information about the reliable messaging *terminate* protocol, see WS-ReliableMessaging: supported specifications and standards. **Attention:** Terminate sequences only if necessary. Refer to the description of the **Delete sequence** button for more information.
Delete sequence	Delete the selected sequences and all their messages, without sending a TerminateSequence message. **Attention:** Delete or terminate sequences only if necessary. If you delete or terminate an active sequence, the resulting messaging behavior is unpredictable and can cause loss of messages. If you are not sure whether you can safely delete or terminate a sequence, do not delete or terminate it; the system automatically deletes sequences that have been inactive for 12 hours.

Outbound sequences settings

This page displays the collection of outbound sequences for the current scope. Outbound sequences are used to transmit messages reliably from the local application to the remote endpoint. Each sequence has a unique identifier.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Outbound sequences** > ***outbound_sequence_name***.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

Runtime tab: Runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General properties

Sequence identifier

This URI is the unique identifier for the sequence.

Required No
Data type Text

Runtime identifier
The identifier that is used within the application server runtime environment.

Required No
Data type Text

Associated application
The application that created the sequence.

Required No
Data type Text

WS-Addressing namespace
The WS-Addressing namespace that is associated with the sequence.

Required No
Data type Text

WS-ReliableMessaging namespace
The namespace that is defined by the version of the WS-ReliableMessaging specification used by the sequence.

Required No
Data type Text

Target endpoint URI
The destination to which messages are transmitted.

Required No
Data type Text

Reply to address
The WS-Addressing replyTo address that is used for WS-ReliableMessaging protocol messages.

Required No
Data type Text

Acknowledgement address
The address used for WS-ReliableMessaging acknowledgements.

Required No
Data type Text

Message depth
The count of messages, held by the reliable messaging layer, that have not yet been transferred.

Required No
Data type Text

Messages sent
The total count of messages sent by the application for this sequence.

Required	No
Data type	Text

Outbound message collection

The messages on the outbound sequence.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Outbound sequences** > ***outbound_sequence_name*** > **[Additional properties] Messages**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

Message number
The index number of the message in the sequence.

State The two possible states are "Sendable" (indicating that it is expected to be sent soon) and "Awaiting sequence initialization".

Buttons

Refresh	Refresh the list of items.
Delete	Delete the selected items.

Message settings

The details of an individual message.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127. At each scope, this panel is available for both inbound and outbound sequences. For example:

- **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Inbound sequences** > ***inbound_sequence_name*** > **[Additional properties] Messages** > ***message_number***
- **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Outbound sequences** > ***outbound_sequence_name*** > **[Additional properties] Messages** > ***message_number***

Runtime tab: Runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General properties

Message number
The index number of the message in the sequence.

Required	No
Data type	Text

Message contents
The contents of the message.

Required	No
Data type	Custom

Inbound sequence collection

This page displays the collection of inbound sequences for the current scope. Each inbound sequence is used to receive messages that have been transmitted reliably.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Inbound sequences**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

The following icons are displayed here and on several other reliable messaging runtime panels:

	**OK**	Everything here, and (if there is a link) in all runtime panels below this link, is running normally.
	**Warning**	Something here, or (if there is a link) in one of the runtime panels below this link, is in an unusual state and you might have to take some action to resolve it. For example, the system might be awaiting a response from an endpoint. In this case, either the response will be received (in which case you need take no action and the runtime information will be updated to "OK") or the reliable messaging destination has stopped acknowledging messages (in which case you have to take some action to resolve the failed sequence).
	**Error**	There is a definite error that you must take some action to resolve, either here or (if there is a link) in one of the runtime panels below this link.

Note that for troubleshooting purposes you only have to follow links to the sub-panels if states other than "OK" are displayed.

Note:

You might see more sequences than you expect, due to sequence reallocation. If a sequence is reallocated, the original and new sequences are both visible.

Sequence identifier
This URI is the unique identifier for the sequence.

Associated application
The application that is receiving messages from the sequence.

Message depth
The count of messages, held by the reliable messaging layer, that have not yet been transferred.

Messages received
The total count of application messages received for the sequence since its creation.

Details
The current state of the sequence. "Connected" indicates that the sequence has been established; "Awaiting redelivery of a missing msg." indicates that there is a gap in the sequence; "Closed" indicates that the sequence has been closed; "Failed due to missing message" indicates that the sequence terminated with a gap in the sequence.

Status
The icon indicates "OK", "Warning" or "Error" as previously described on this page. A more precise indication of the sequence state is given in the "Details" column.

Buttons

Dispatch messages to application	Dispatch the messages on the selected sequences to the associated applications.
Export undispatched messages	Export the messages from the selected sequences to compressed files. If you select any sequences that have no messages to export, a warning or error message is displayed.
Close sequence	Close the selected sequences and send a CloseSequence message to indicate that the sequence is complete and will not be sending any further messages. The *close* protocol does not delete the resources for the sequence, so you can still access any undispatched or unsent messages. If you are using Version 1.0 of the WS-ReliableMessaging specification (which does not support this option) an error message is displayed. For more information about the reliable messaging *close* protocol, see WS-ReliableMessaging: supported specifications and standards.
Terminate sequence	Close the selected sequences and send a TerminateSequence message to indicate that the sequence is complete and will not be sending any further messages. The *terminate* protocol deletes all resources for the sequence. For more information about the reliable messaging *terminate* protocol, see WS-ReliableMessaging: supported specifications and standards. **Attention:** Terminate sequences only if necessary. Refer to the description of the **Delete sequence** button for more information.

Delete sequence and messages	Delete the selected sequences and all their messages. Use this in cases where the remote endpoint is unable to respond to any reliable messaging protocol messages, and therefore the sequence cannot be closed or terminated. **Attention:** Delete or terminate sequences only if necessary. If you delete or terminate an active sequence, the resulting messaging behavior is unpredictable and can cause loss of messages. If you are not sure whether you can safely delete or terminate a sequence, do not delete or terminate it; the system automatically deletes sequences that have been inactive for 12 hours.

Inbound sequences settings

This page displays the collection of inbound sequences for the current scope. Each inbound sequence is used to receive messages that have been transmitted reliably.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Inbound sequences** > ***inbound_sequence_name***.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

Runtime tab: Runtime properties for this object. These properties directly affect the current runtime environment, but are not preserved when that environment is stopped. To preserve runtime property values, change the equivalent property values on the Configuration tab. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General properties

Sequence identifier

This URI is the unique identifier for the sequence.

Required	No
Data type	Text

Associated application

The application that is receiving messages from the sequence.

Required	No
Data type	Text

WS-Addressing namespace

The WS-Addressing namespace that is associated with the sequence.

Required	No
Data type	Text

WS-ReliableMessaging namespace

The namespace that is defined by the version of the WS-ReliableMessaging specification used by the sequence.

Required	No

Data type Text

Target endpoint URI
The destination to which messages are transmitted.

Required No
Data type Text

Reply to address
The WS-Addressing replyTo address that is used for WS-ReliableMessaging protocol messages.

Required No
Data type Text

Acknowledgement address
The address used for WS-ReliableMessaging acknowledgements.

Required No
Data type Text

Message depth
The count of messages, held by the reliable messaging layer, that have not yet been transferred.

Required No
Data type Text

Highest inbound message number
The highest message number that has been received within the sequence.

Required No
Data type Text

In-order delivery
When this parameter is true, messages are dispatched to the application in the order that they were assigned to the sequence.

Required No
Data type Text

Inbound message collection
The messages on the inbound sequence.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Inbound sequences** > ***inbound_sequence_name*** > **[Additional properties] Messages**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

Message number
The index number of the message in the sequence.

State The only possible state is "Awaiting dispatch to application".

Buttons

Refresh	Refresh the list of items.
Delete	Delete the selected items.

Acknowledgement state collection
The ranges of message sequence numbers received from the WS-ReliableMessaging source. If more than one range is displayed, this indicates a gap in the messages received. If "In-order delivery" is selected for the sequence manager, messages with a sequence number greater than the lowest gap cannot be delivered to the application until the gap is closed.

To view this page in the console, click one of the paths to this panel. For example **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Inbound sequences** > ***inbound_sequence_name*** > **[Additional properties] Ack state**.

To browse or change the properties of a listed item, select its name in the list.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127.

Low end of range
The message number for the lowest numbered message in this range.

High end of range
The message number for the highest numbered message in this range.

Buttons

Refresh	Refresh the list of items.

Export messages settings
Export the messages from the selected sequences to ZIP files.

The WS-ReliableMessaging runtime panels are available at many different scopes within the administrative console. For more information, see “WS-ReliableMessaging - administrative console panels” on page 3127. At each scope, this panel is available for both inbound and outbound sequences. For example:

- **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Inbound sequences** > **[Button bar] Export undispatched messages**

- **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Additional Properties] Reliable messaging state** > **Runtime** > **Outbound sequences** > **[Button bar] Export unsent messages**

General properties

Export messages
A list of the exported messages ZIP files that have been created. To save these ZIP files to your file system, click each file individually then use your web browser "save file" option.

WS-Notification Service client settings

Use this page to manage policy sets and bindings or to access additional information for this WS-Notification service client.

To view this page in the console, click the following path:

Services > **Service clients** > ***ws-notification_service_client_name***

This page shows the policy set and binding configuration information for a single Version 7.0 WS-Notification service client (that is, a service client using the JAX-WS rather than a JAX-RPC WS-Notification implementation), and also gives you links to the associated service integration bus and WS-Notification service.

Note: The same policy set and binding configuration information can also be viewed and modified through the “Service client policy sets and bindings” page, which gives an upper-level view of all service clients associated with a particular WS-Notification service. Through the “Service client policy sets and bindings” page, you can assign a policy set or binding to a specific service client, or to all clients for the service. To view the “Service client policy sets and bindings” page for a given WS-Notification service, click one of the following paths:

- **Service integration** > **WS-Notification** > **Services** > ***service_name*** > **[Additional properties] Outbound request policy sets and bindings**
- **Service integration** > **Buses** > ***bus_name*** > **[Services] WS-Notification services** > ***service_name*** > **[Additional properties] Outbound request policy sets and bindings**

Configuration tab: Configuration properties for this object. These property values are preserved even if the runtime environment is stopped then restarted. See the information center task descriptions for information about how to apply configuration changes to the runtime environment.

General Properties

Service client
Specifies the name of the service client that is displayed.

Additional Properties

Service integration bus
The system integration bus hosting this WS-Notification service web service client.

WS-Notification service
The WS-Notification service hosting this client.

Policy Set Attachments

This section allows you to attach a policy set to the WS-Notification service client. You can complete the attachment by providing system-specific configuration when you assign the appropriate binding.

For more information about the policy set attachment options, see “Service client settings” on page 2646.

Note: For WS-Notification service clients, policy set attachments are not supported at the endpoint (port) or operation level. Therefore, endpoints or operations are not selectable, and they are shown as inheriting any policy set or binding that is attached to the service client.

To act on one or more of the listed items, select the check boxes next to the names of the items that you want to act on, then use the buttons provided.

To change what entries are listed, or to change what information is shown for entries in the list, use the Filter settings.

About policy set bindings

In this version of the product there are two types of bindings, application-specific bindings and general bindings.

Application specific binding

You can create application-specific bindings only at a policy set attachment point. These bindings are specific to and constrained to the characteristics of the defined policy. Application specific bindings are capable of providing configuration for advanced policy requirements, such as multiple signatures; however, these bindings are only reusable within an application. Furthermore, application-specific bindings have very limited reuse across policy sets.

When you create an application-specific binding for a policy set attachment, the binding begins in a completely unconfigured state. You must add each policy, such as WS-Security or HTTP transport, that you want to override the default binding and fully configure the bindings for each policy that you have added. For WS-Security policy, some high level configuration attributes such as TokenConsumer, TokenGenerator, SigningInfo, or EncryptionInfo might be obtained from the default bindings if they are not configured in the application-specific bindings.

For service providers, you can only create application-specific bindings by selecting **Assign Binding > New Application Specific Binding** for service provider resources that have an attached policy set. See service providers policy sets and bindings collection. Similarly, for service clients, you can only create application-specific bindings by selecting **Assign Binding > New Application Specific Binding** for service client resources that have an attached policy set. See service client policy set and bindings collection.

General bindings

These bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they are however not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings:

- General provider policy set bindings
- General client policy set bindings

You can create general provider policy set bindings by accessing **Services > Policy sets > General provider policy set bindings** > **New** in the general provider policy sets panel or by accessing **Services > Policy sets > General client policy set bindings** > **New** in the general client policy set and bindings panel. See “Defining and managing service client or provider bindings” on page 2685. General provider policy set bindings might also be used for trust service attachments.

Buttons

Attach Client Policy Set	View a list of policy sets available for attachment to the selected WS-Notification service client. To attach the policy set to the selected service client, select a policy set from the drop-down list. To close the list, click **Attach Client Policy Set**.
Detach Client Policy Set	Detach a policy set from a selected WS-Notification service client. After the policy set is detached, if there is no policy set attached at the upper level (that is, at the level of the WS-Notification service), the Attached Client Policy Set column displays `None` and the Binding column displays `Not Applicable`. If there is a policy set attached at the level of the WS-Notification service, the Attached Client Policy Set column displays *`policy_set_name`*`(inherited)` for all endpoints and operations. Similarly, for endpoints and operations, the binding name is displayed with `(inherited)` after it.
Assign Binding	Select from a list of available bindings for the selected policy set attachment. All the bindings are listed along with the following options: • Default • New Application Specific Binding For more information about these options, see “Service client settings” on page 2646. To close the drop-down list, click Assign Binding.

Chapter 35. Administering web services - RESTful services

You can use Java™ API for RESTful Web Services (JAX-RS) to develop services that follow Representational State Transfer (REST) principles. RESTful services are based on manipulating resources. Resources can contain static or dynamically updated data. By identifying the resources in your application, you can make the service more useful and easier to develop.

Planning JAX-RS web applications

Planning to use JAX-RS to enable RESTful services

By using the Java API for RESTful Web Services (JAX-RS) API, application developers can quickly develop RESTful applications. When planning to use JAX-RS to enable RESTful services, consider how to best implement the capabilities and characteristics of a RESTful application with JAX-RS.

Before you begin

Read the overview of JAX-RS information to learn about REST services and the advantages of using JAX-RS to build RESTful services.

About this task

JAX-RS is a programming model that provides a mechanism for developing services that follow Representational State Transfer (REST) principles. Using JAX-RS, development of RESTful services is simplified.

JAX-RS is a Java API for developing REST applications quickly. While JAX-RS provides a faster way for developing web applications than servlets, the primary goal of JAX-RS is to build RESTful services. JAX-RS 1.0 defines a server-side component API to build REST applications. IBM JAX-RS provides an implementation of the JAX-RS (JSR 311) specification.

By using JAX-RS technology, REST applications are simpler to develop, simpler to consume, and simpler to scale when compared to other types of distributed systems. Many popular and widely used Internet services have successfully provided RESTful APIs to their applications. Third parties have used various REST APIs to build their own businesses and applications.

Due to the simple consumption of RESTful services, you can write clients in many languages on different platforms. Most languages require no third-party libraries as long as there is a method to use an HTTP connection. Because of the pervasiveness of web browsers, the most prevalent clients are typically web browsers. For example, many Web 2.0 properties use a JavaScript framework such as Dojo toolkit for developing a client in a browser in conjunction with a RESTful server-side application that provides the data for the client.

Procedure

1. Review existing business and middleware applications in your environment to determine which services you want to implement as REST services.
2. Define the resources in your RESTful applications.
3. Determine the URL patterns, operations, and media type formats to use for each resource.
   a. Define the URI patterns for resources in RESTful applications.
   b. Define the client capabilities for RESTful applications using HTTP methods .
   c. Define the HTTP headers and response codes for RESTful applications using HTTP methods.

Results

You have a design plan for using JAX-RS to implementing REST services.

Defining the resources in RESTful applications

You can use Java API for RESTful Web Services (JAX-RS) to develop services that follow Representational State Transfer (REST) principles. RESTful services are based on manipulating resources. Resources can contain static or dynamically updated data. By identifying the resources in your application, you can make the service more useful and easier to develop.

Before you begin

After you have identified the application that you want to expose as a RESTFUL service, you must first define the resources for your RESTful application. When defining the resources for your application, consider the type of data do you want to expose. Perhaps you already have a relational database that contains information that you want to expose to users using REST technology. Do you already have a set of Java classes defined for accessing that data?

For example, consider the case of an application defined to support a book store. This application currently has a database with several tables that define the various items in the collection of books and the inventory of each book. In this example, there are a number of ways to represent the data in the database in a RESTful application. One approach is to consider each table as an individual resource, so that each of the verbs in the RESTful request maps to the actions that the database supports on that table such as select, insert, update, delete. This example is a simple approach to creating a RESTful application. This approach using the book store example is also used in the documentation that describes defining URL patterns for resources, resource methods, HTTP headers and response codes, media types, and parameters for request representations to resources

In support of this database for the book store application, there might already be existing code that is responsible for accessing the database and retrieving the data from each table. Even though the rows in each of the tables logically represents each resource, the accessor classes are used to define the resources. The implementing JAX-RS applications documentation provides more details on how these classes are incorporated into your JAX-RS application.

Alternately, you might have more static content that does not reside in a database that you want to distribute as resources. Whether it is a collection of documents in various formats or a resource-based facade for other remote systems, using JAX-RS, you can distribute content from multiple sources.

About this task

Resources are the basic building block of a RESTful service. Examples of a resource from an online book store application include a book, an order from a store, and a collection of users.

Resources are addressable by URLs and HTTP methods can perform operations on resources. Resources can have multiple representations using different formats such as XML and JSON. You can use HTTP headers and parameters to pass additional information that is relevant to the request and response.

With JAX-RS, you can annotate existing or new Plain Old Java objects (POJO) with JAX-RS specific annotations. JAX-RS annotated resource classes and the annotated methods are invoked depending on the URI patterns. You can use the annotated resource classes after these resource classes are added to the list of resources returned by the overridden methods in the JAX-RS application class.

Procedure

1. Identify the types of resources in the application.
2. (optional) Identify existing Java classes that you can use as resource classes.

3. Create new Java classes for resources that do not have an existing Java class.

Results

You have defined the content that you want to expose as a collection of resources in your application.

What to do next

Based on the resources that you have defined, read about defining URL patterns for resources, resource methods, HTTP headers and response codes, media types, and parameters for request representations to resources to learn more about additional steps you can take to define the resources for your JAX-RS application.

Defining the URI patterns for resources in RESTful applications

Representational State Transfer (REST) services are based on manipulating resources. Resources for RESTful services are addressable, and URLs are the primary way of achieving addressability in REST.

Before you begin

Identify the resources in the application that you want to expose as a RESTful service.

About this task

URLs are used to specify the location of a resource. Interaction between the server and client is based on issuing HTTP operations to URLs. Defining URL patterns is important because URLs often have a long lifetime so that clients can directly address a resource long after the resource is initially discovered.

URLs are typically used when you enter addresses to web browsers, such as `http://www.ibm.com/` or `http://www.example.com/bookstore/books/ISBN123`. Although URLs are not required to be understandable by users, RESTful services that provide logical URLs in understandable patterns enable client application developers to work efficiently.

RESTful clients use URLs to manipulate resources. Each resource must have its own unique URL. Some URL patterns have a collection path with a unique identifier appended. For example, you can use `http://www.example.com/bookstore/books` as the collection resource URL, `http://www.example.com/bookstore/books/ISBN123` as a unique book resource URL, and you can use `http://www.example.com/bookstore/books/ISBN123/authors` to retrieve a collection resource describing `ISBN123` authors.

The application developer must carefully consider the granularity of URLs because it can affect usage of the application and performance. For example, you can include the author information of a book as part of the book resource or you can define the author information as a unique resource with its own URL that is referenced in the book resource. Depending on the reuse of resources, it might be more efficient to define a separate resource for the author information that is referenced in a hyperlink of the book resource for cases when the author writes a different book.

After an initial URL is given to a client, subsequent related requests are discoverable by parsing the current resource. In the book example, a GET request to `http://www.example.com/bookstore/books/` retrieves a list of book URLs that can include `http://www.example.com/bookstore/books/ISBN123`.

Because systems rely on resources being available, URLs typically have longevity. Because HTTP has built-in status codes for redirection, such as the 301 moved permanently code and the 307 temporarily redirected code, users and clients with caches often reuse previously discovered URLs first. You can additionally consider including a version identifier in the URL pattern, such as `http://www.example.com/bookstore/v2/books/ISBN123`. Like the planning involved to define an interface using Java code, be sure to carefully choose your URL patterns because of expected longevity.

In Java API for RESTful Web Services (JAX-RS), you must add @Path annotations to the Java class files or the Java methods to define the relative URL of the resource. You can use JAX-RS subresource locators and subresource methods to define resources. Use parameters, such as the path parameter or matrix parameter, in the URL to identify the resource.

The value in the @Path annotation defines the relative part of the full URL to your resource. The base URL is derived from the domain, port, application module context root, and any URL pattern mappings in the web.xml file of the application module. For example, if the domain is `www.example.com`, the port is `9060`, the module context root is `example`, the servlet URL pattern is `store/*`, and the value of the @Path annotation is `/bookstore/books`. The full URL is: `http://www.example.com:9060/example/store/bookstore/books`.

Procedure

1. Identify the types of resources in the application. Suppose that you have two types of resources, a BooksCollection and an individual Book object which have the following class definitions:

```
public class BooksCollection {
    public BooksCollection() {
        /* no argument constructor */
    }

}

public class Book {
    public Book(String ISBN) {
        /* This constructor has an argument that will be annotated with a JAX-RS annotation.
         See the JAX-RS specification for information on valid constructors. */
    }
}
```

   As defined in the JAX-RS specification, by default, resource instances are created per request. For the JAX-RS runtime environment to create a resource instance, you must have either a constructor with no argument or a constructor with only JAX-RS annotated parameters present.

2. Add a @javax.ws.rs.Path annotation to each resource class. For each @javax.ws.rs.Path annotation, set the value as the part of the URL after the base URL of the application.

```
/*
 * BooksCollection.java
 * This Java class represents the books collection URL at /bookstore/books.
 */
@javax.ws.rs.Path("/bookstore/books/")
public class BooksCollection {

}
```

   After completing the application, you can use the resource by visiting `http://<host_name>:<port>/<context_root>/<servlet_path>/bookstore/books`. For this URL, specify the context root value as the part of the URL after the context module. Specify the servlet path as any URL patterns in the web.xml file, if it exists.

3. (optional) Determine if a resource needs to use part of the URL as a parameter. If a resource needs to use part of the URL as a parameter, such as an identifier, you can use the @javax.ws.rs.Path annotation with a regular expression. You can then add a @javax.ws.rs.PathParam annotation in either the resource constructor or the resource method.

```
/*
 * Book.java represents individual books.
  */
@javax.ws.rs.Path("/bookstore/books/{bookID}")
public class Book {
   public Book(@javax.ws.rs.PathParam("bookID") String ISBN) {

   }
}
```

   When an HTTP request is made to `http://<host_name>:<port>/<context_root>/<servlet_path>/bookstore/books/ISBN_number`, a Book instance is created with *ISBN_number* passed in to the ISBN parameter of the constructor.

For more information about other possible parameters, read about defining parameters for requests to resources in RESTful applications.

4. Create the javax.ws.rs.core.Application subclass to define to the JAX-RS runtime environment which classes are part of the JAX-RS application. The resource classes are returned in the getClasses() method; for example:

```
public class BookApplication extends javax.ws.rs.core.Application {
    public Set<Class<?>> getClasses() {
        Set<Class<?>> classes = new HashSet<Class<?>>();
        classes.add(BooksCollection.class);
        classes.add(Book.class);
        return classes;
    }
}
```

By defining the javax.ws.rs.core.Application subclass, classes returned from its methods are registered to the JAX-RS runtime environment. When configuring the web.xml file, you must specify the javax.ws.rs.core.Application subclass as a parameter to the servlet or filter. For more information, read about configuring the web.xml file for JAX-RS applications.

Results

You have created a URL to identify your resources for your RESTful service. By considering issues with URL patterns early in the application design, the RESTful service increases its usability and value over an extended time.

What to do next

The resource at the defined URL exists. However, the resource does not yet have any methods to handle HTTP method actions such as GET, POST, PUT, or DELETE. See the defining resource methods for RESTful applications to learn more about defining capabilities of resources using supported HTTP methods.

Defining resource methods for RESTful applications

Individual resources can define their capabilities using supported HTTP methods. In Representational State Transfer (REST) services, the supported methods are GET, PUT, DELETE, and POST. All operations are typically conducted by using one of the predefined HTTP methods with a resource.

Before you begin

Understand the predefined HTTP methods and their known attributes. Some HTTP methods are meant to be safe, meaning that issuing the request does not change the resource, or idempotent, meaning that multiple invocations of the operation do not change the result. While HTTP methods are defined to have certain attributes, the service implementation follows the definitions. See the HTTP method definitions information to learn more about the common set of methods for HTTP.

About this task

Clients use HTTP methods to perform certain operations. Unlike other distributed systems where unique interfaces are defined by each system, RESTful systems based on HTTP mainly rely on predefined methods. The four most common methods are GET, PUT, DELETE, and POST. Resources are not required to permit all HTTP methods for all clients.

The HTTP GET method retrieves a resource representation. It is safe and idempotent. Repeated GET requests do not change any resources.

The HTTP PUT method is often used to update resources or to create a new entity at a known URL. When a resource must be updated or created, an HTTP PUT method is issued at the resource URL with the new resource data as the request entity, also known as the message body. The HTTP PUT method is

idempotent so multiple identical PUT requests with the same entity to the same URL yields the same result as if only one PUT request was issued. This method assumes that no other relevant requests were made.

The HTTP DELETE method removes a resource at a given URL. It is also idempotent.

The HTTP POST method is often used when creating a resource in a collection when the final resource URL is not known. For instance, an administrator issues a POST request to a `/users` collection resource that creates a user with a unique ID such as `1234567890`. The unique ID is then used as part of the URL path to describe the new user resource, such as `/users/1234567790`. It is not safe and is not idempotent. In this case, the multiple POST requests to the `/users` collection can continue creating a new unique ID and adding this new ID to the users collection even if the user has the same information.

Because most RESTful services use well-known HTTP methods that provide expected results, you can more easily create clients. RESTful service developers can take advantage of the expected behaviors of HTTP methods. Resource methods can also use parameters, such as path parameters, query parameters, or matrix parameters to identify the resource. Read about defining the use of parameters for request representations to resources to learn more.

(optional) If you have a sub-resource method and a sub-resource locator method that have an `@Path` annotation with the same value in the same resource class, the sub-resource locator is not considered when determining the method to invoke by default. This is in compliance with the JAX-RS specification.

Use the `wink.searchPolicyContinuedSearch` property with a value of `true` to modify this behavior. This results in sub-resource locators being used if no sub-resource methods match the request.

To enable the property, include a properties file in the `WEB-INF` directory of the module that has the `wink.searchPolicyContinuedSearch` property and value specified. In the `web.xml` file of the application module, include an `init-param` element with the `propertiesLocation` value for the `param-name` element. The `param-value` element specifies the location of the properties file, for example, `WEB-INF/my_wink.properties`.

The following example illustrates the `web.xml` file:

```
<servlet>
    ...
    ...
    <init-param>
        <param-name>propertiesLocation</param-name>
        <param-value>/WEB-INF/my_wink.properties</param-value>
    </init-param>
</servlet>
```

The following example illustrates the `WEB-INF/my_wink.properties` properties file:

```
wink.searchPolicyContinuedSearch=true
```

Procedure

1. Identify the types of resources in the application.

   For each resource class, identify an existing method or create a method that you want to invoke for each supported HTTP method. Methods that respond to HTTP requests are also known as resource methods. For each resource method in the resource class, annotate the Java method with a single JAX-RS HTTP method annotation such as @javax.ws.rs.GET, @javax.ws.rs.POST, @javax.ws.rs.DELETE or @javax.ws.rs.PUT. For example, if an HTTP GET method is supported by the BooksCollection class, then you can create and annotate a method like the following snippet:

```
@javax.ws.rs.Path("/bookstore/books/")
public class BooksCollection {
    @javax.ws.rs.GET
    public String getBooksCollection() {
        String str = /* Construct a String representation of the resource. */
        return str;
    }
}
```

When issuing an HTTP GET request to `http://<host_name>:<port>/<context_root>/<servlet_path>/bookstore/books` URL using a web browser or another HTTP client, the previous getBooksCollection() method is invoked.

2. Ensure that the resource supports the required HTTP methods.

   Each resource typically has multiple resource methods; for example:

```
@javax.ws.rs.Path("/bookstore/books/{bookID}")
public class Book {
    /* This is a database key. */
    private String ISBN;

    public Book(@javax.ws.rs.PathParam("bookID") String ISBN) {
        this.ISBN = ISBN;
    }

    @javax.ws.rs.GET
    public String retrieveSpecificBookInformation() {
        /* This code retrieves a book resource based on the ISBN key. */
    }

    @javax.ws.rs.PUT
    public String updateBookInformation(String updatedBookInfo) {
        /* This code updates the book resource based on ISBN key and the entity (message body) sent
         in the request that is stored in updatedBookInfo. */
    }

    @javax.ws.rs.DELETE
    public void removeBook() {
        /* This code deletes a book resource based on ISBN key. */
    }
}
```

   When issuing an HTTP GET request to the `http://<host_name>:<port>/<context_root>/<servlet_path>/bookstore/books/<isbn_number>` URL using a web browser or another HTTP client, the retrieveSpecificBookInformation() method is invoked. Sending an HTTP PUT request to the same URL invokes the updateBookInformation method and any content in the request message body is passed as the value of the `updatedBookInfo` object. Finally, sending an HTTP DELETE request to the same URL invokes the removeBook() method.

   Note: According to the JAX-RS specification, you must not put multiple HTTP method annotations, such as @javax.ws.rs.POST or @javax.ws.rs.PUT on the same resource method Because HTTP methods have uniquely defined semantics, do not use a resource method for multiple HTTP methods.

Results

You have defined valid operations for the resources.

Defining the HTTP headers and response codes for RESTful applications

HTTP headers and status codes are useful to help intermediary and client programs understand information about requests and responses for applications. HTTP headers contain metadata information. HTTP status codes provide status information about the response.

Before you begin

See the HTTP 1.1 specification to become familiar with HTTP headers and HTTP status codes.

About this task

HTTP headers contain metadata information such as security authentication information, the user agent that is used, and cache control metadata. Standard HTTP headers are defined in the HTTP specification; however, you can use custom HTTP headers, if necessary.

You can read HTTP headers from the request and set the headers in the response. There are a set of common request and response headers, but there are also unique request and unique response headers. JAX-RS provides the HttpHeaders injectable interface and the @HeaderParam parameter annotation for reading HTTP headers. If a javax.ws.rs.core.Response object is returned from a resource method, you can set HTTP headers on the response. Also, you can set HTTP headers when an entity is written using the MessageBodyWriter interface.

You can set HTTP response status codes to help client programs understand the response. While responses can contain an error code in XML or other format, client programs can quickly and more easily understand an HTTP response status code. The HTTP specification defines several status codes that are typically understood by clients.

Procedure

- To read a specific request header, add a @javax.ws.rs.HeaderParam annotated parameter.

```
@javax.ws.rs.Path("/bookstore/books/{bookID}")
public class Book {
    @javax.ws.rs.GET
    public String retrieveSpecificBookInformation(@javax.ws.rs.HeaderParam("User-Agent") String theUserAgent) {
        /* The book ID was sent in a HTTP request header with the name "bookID". */
    }
}
```

- To read any request header, use the @javax.ws.rs.core.Context javax.ws.rs.core.HttpHeaders injected object.

```
@javax.ws.rs.Path("/bookstore/books/{bookID}")
public class Book {
    @javax.ws.rs.GET
    public String retrieveSpecificBookInformation(@javax.ws.rs.core.Context HttpHeaders requestHeaders) {
        /* Call methods on "requestHeaders" to get any request header sent by the client. */
        List<String> bookIdValues = requestHeaders.getRequestHeader("User-Agent");
    }
}
```

- To set a response status code or header, return a javax.ws.rs.core.Response object and build the response with the appropriate status code and headers.

```
@javax.ws.rs.Path("/bookstore/books/{bookID}") public class Book {
    @javax.ws.rs.GET public javax.ws.rs.core.Response retrieveSpecificBookInformation() {
        return Response.status(200).header("responseHeaderName", "responseHeaderValue").header("anotherResponseHeaderName", "foo").build();
    }
}
```

Results

You have used HTTP headers to read request headers and set response status codes and headers for JAX-RS web applications.

Defining media types for resources in RESTful applications

Resources are represented by multiple formats. XML, JavaScript Object Notation (JSON), Atom, binary formats such as PNG, JPEG, GIF, plain text, and proprietary formats are used to represent resources. Representational State Transfer (REST) provides the flexibility to represent a single resource in multiple formats.

Before you begin

Define the resources in the JAX-RS web application.

About this task

Resources have representations. A resource representation is the content in the HTTP message that is sent to, or returned from the resource using the URI. Each representation that a resource supports has a corresponding media type. For example, if a resource is going to return content formatted as XML, you can use `application/xml` as the associated media type in the HTTP message.

Depending on the requirements of your application, resources can return representations in a preferred single format or in multiple formats. For example, resources that are accessed using JavaScript clients might prefer JSON representations because JSON is easy to consume.

JAX-RS provides @Consumes and @Produces annotations to declare the media types that are acceptable for a resource method to read and write.

JAX-RS also maps Java types to and from resource representations using entity providers. A `MessageBodyReader` entity provider reads a request entity and deserializes the request entity into a Java type. A `MessageBodyWriter` entity provider serializes from a Java type into a response entity.

Table 271. Standard entity providers and basic Java types. This table includes the standard entity providers and basic Java types that are included in the JAX-RS runtime environment, along with the corresponding supported content types.

Java type	MessageBodyReader	MessageBodyWriter	Supported Content types
byte[]	X	X	*/*
java.io.InputStream	X	X	*/*
java.io.Reader	X	X	*/*
java.lang.String	X	X	*/*
java.io.File	X	X	*/*
javax.activation.DataSource	X	X	*/*
javax.xml.transform.Source	X	X	text/xml, application/xml, application/*+xml
javax.ws.rs.core.MultivaluedMap	X	X	application/x-www-form-urlencoded
JAXB types	X	X	text/xml, application/xml, application/*+xml
javax.ws.rs.core.StreamingOutput		X	*/*

If a `String` value is used as the request entity parameter, the `MessageBodyReader` entity provider deserializes the request body into a new `String`. If a JAXB type is used as the return type on a resource method, the MessageBodyWriter serializes the Java Architecture for XML Binding (JAXB) object into a response body.

If you need a custom mapping from a Java type to a specific representation, see the information for using an application-defined entity provider.

If your client can handle multiple formats and you want the server to determine the best resource representation to return, read about using content negotiation in JAX-RS applications to serve multiple content types.

The specifications for XML, JSON, and Atom provide details regarding the formats of resource representations for applications. See the specifications to learn more about the formats of resource representations.

Procedure

1. Determine the resource representation format such as XML, JSON, or ATOM to use for either the request or the response.
2. Add the @Consumes and @Produces annotations appropriately to the resource method.
3. If the resource needs to read the content of the request, add a request entity parameter to the resource method. The request entity parameter is a single Java parameter on the method that does not have an annotation.

4. If the resource method returns content in the response, return a Java object that is writable by a JAX-RS entity provider. This Java object is mapped to the response entity in the HTTP response. The returned object must be a JAX-RS supported Java type or wrapped in a javax.ws.rs.core.Response or javax.ws.rs.core.GenericEntity type.

Results

You have mapped the request entities to the resource method entity parameter and any response objects that are returned are mapped to the response entity for the resource representation.

Example

The following example illustrates defining XML as the resource representation for a RESTful bookstore application.

1. Identify the resource methods that you want to read the request entity or return a response entity.

   In the retrieveSpecificBookInformation method example that follows, there is no request entity that is read. However, there is a response object that is returned. This object wraps a JAXB object that contains the entity information. Adding the @Produces annotation on the resource method with a media type of `application/xml` indicates that the resource method always returns an XML representation with a media type of `application/xml`.

   Clients that have an Accept HTTP request header value compatible with the `application/xml` media type invoke the method correctly.

   Clients that do not have an Accept HTTP header value compatible with the `application/xml` media type automatically receive a `406 Not Acceptable` response status which indicates that the server cannot produce an acceptable response.

   The following example identifies the resource methods that read the request entity or return a response entity:

```
/*
 * Book.java
 * This class represents individual books. The @Produces annotation specifies a media type of application/xml.
 */
@Path("/bookstore/books/{bookID}")
public class Book {
  @GET
  @Produces("application/xml")
  public javax.ws.rs.core.Response retrieveSpecificBookInformation(@PathParam("bookID") String theBookID,
   @Context javax.ws.rs.core.HttpHeaders headers) {
   /* ... */
   return
    Response.ok(/* JAXB object to represent response body entity */).expires(/* Expires response header value*/).header("CustomHeaderName", "CustomHeaderValue").build();
   }
   @PUT
   public String updateBookInformation(@PathParam("bookID") String theBookID, String theRequestEntity,
    @javax.ws.rs.HeaderParam("Content-Length") String contentLengthHeader) { /* ... */ }

   @DELETE
   public void removeBook(@PathParam("bookID") String theBookID) { /* ... */ }
}
```

2. Identify the resource methods that need to consume the request information.

   In the following snippet, the PUT method on the book resource accepts the request entity content if a media type of `text/plain` is sent, as defined in the @Consumes annotation. This method returns content with a `text/plain` representation as specified in the @Produces annotations.

   If a client does not send a message with a Content-Type value of `text/plain`, then the PUT resource method is not invoked. If `Content-Type: application/xml` is sent in the HTTP request headers, the `updateBookInformation` Java method is not be called.

   The DELETE method neither reads a request entity nor returns a response entity; therefore, it does not require either an @Consumes or an @Produces annotation.

   The following example identifies the resource methods that consume the request information:

```
/*
 * Book.java
 * This class represents represent individual books with custom headers.
 */
@Path("/bookstore/books/{bookID}")
public class Book {
  @GET
  @Produces("application/xml")
  public javax.ws.rs.core.Response retrieveSpecificBookInformation(@PathParam("bookID") String theBookID, @Context javax.ws.rs.core.HttpHeaders headers) {
```

```
    /* ... */
    return Response.ok(/* JAXB object to represent response body entity */).expires(/* Expires response header value).header("CustomHeaderName", "CustomHeaderValue").build();
  }
  @PUT
  @Consumes("text/plain")
  @Produces("text/plain")
  public String updateBookInformation(@PathParam("bookID") String theBookID, String theRequestEntity, @javax.ws.rs.HeaderParam("Content-Length") String contentLengthHeader) {
    /* ... */
    String responseEntity = /* a plain text representation */;
    return responseEntity;
  }

  @DELETE
  public void removeBook(@PathParam("bookID") String theBookID) { /* ... */ }
}
```

What to do next

See the JAX-RS specification for a list of all the standard media formats that are supported for representations.

Advanced users might consider defining custom mappings of Java types to representations or using content negotiation for clients to negotiate preferred resource representations. To learn more about these options, see the using custom defined entity formats information or the serving multiple content types with content negotiation information.

Defining parameters for request representations to resources in RESTful applications

Parameters are used to pass and add additional information to a request. You can use parameters as part of the URL or in the headers. Path parameters, matrix parameters, query parameters, header parameters, and cookie parameters are useful for passing in additional information to a request.

About this task

Multiple parameters types exist. Java API for RESTful Web Services (JAX-RS) enables easy access to all the types of parameters using annotated injected parameters.

You can use any basic Java primitive type including java.lang.String as parameters, as well as Java types with a constructor that uses a single String or a valueOf(String) static method. Additionally, you can use `List`, `SortedSet`, and `Set` interfaces where the generic type is one of the previously mentioned types, such as a Set when a parameter can have multiple values. If you need to parse requests, then use String as the parameter type to enable you to complete basic inspection and customization of error path responses.

JAX-RS provides the following annotations to use on resource method parameters to specify that the resource method can be invoked with correct parameter values.

javax.ws.rs.PathParam annotation

Path parameters are part of the URL. For example, the URL can include `/collection/{item}`, where `{item}` is a path parameter that identifies the item in the collection. Because path parameters are part of the URL, they are essential in identifying the request.

If parts of the URL are parameters, you can use a @javax.ws.rs.PathParam annotated parameter; for example:

```
@javax.ws.rs.Path("/bookstore/books/{bookId}")
public class BooksCollection {
    @javax.ws.rs.GET
    public String getSpecificBookInfo(@javax.ws.rs.PathParam("bookId") String theBookId) {
        /* theBookId would contain the next path segment after /bookstore/books/  */
    }
}
```

In this example, requests to `/bookstore/books/12345` assigns the value of `12345` to the `theBookId` variable.

javax.ws.rs.MatrixParam annotation

Matrix parameters are part of the URL. For example, if the URL includes the path segment, `/collection;itemID=itemIDValue`, the matrix parameter name is `itemID` and `itemIDValue` is the value.

You can read matrix parameters with a @javax.ws.rs.MatrixParam annotated parameter; for example:

```
@javax.ws.rs.Path("/bookstore/books")
public class BooksCollection {
    @javax.ws.rs.GET
    public String getBookCollectionInfo(@javax.ws.rs.MatrixParam("page") int page, @javax.ws.rs.MatrixParam("filter") String filter) {
        /* This statement uses the page and filter parameters. */
    }
}
```

In this example, requests to`/bookstore/books;page=25;filter=test` invoke the `getBookCollectionInfo` parameter so that the value for the `page` variable is set to `25` and the value for `filter` variable is set to `test`.

javax.ws.rs.QueryParam annotation

Query parameters are appended to the URL after a "?" with name-value pairs. For instance, if the URL is `http://example.com/collection?itemID=itemIDValue`, the query parameter name is `itemID` and `itemIDValue` is the value. Query parameters are often used when filtering or paging through HTTP GET requests.

You can read query parameters with a @javax.ws.rs.QueryParam annotated parameter; for example:

```
@javax.ws.rs.Path("/bookstore/books")
public class BooksCollection {
    @javax.ws.rs.GET
    public String getBookCollectionInfo(@javax.ws.rs.QueryParam("page") int page, @javax.ws.rs.QueryParam("filter") String filter) {
        /* This statement uses the page and filter parameters. */
    }
}
```

In this example, requests to`/bookstore/books;page=25;filter=test` invoke the `getBookCollectionInfo` parameter so that the value for the `page` variable is set to `25` and the value for `filter` variable is set to `test`.

javax.ws.rs.HeaderParam annotation

Header parameters are HTTP headers. While there are pre-defined HTTP headers, you can also use custom headers. Headers often contain control metadata information for the client, intermediary, or server.

If a HTTP request header must be read, use the @javax.ws.rs.HeaderParam annotation; for example:

```
@javax.ws.rs.Path("/bookstore/books/")
public class BooksCollection {
    @javax.ws.rs.GET
    public String getBookCollectionInfo(@javax.ws.rs.HeaderParam("Range") String rangeValue) {
        /* The rangeValue variable contains the value of the HTTP request header "Range" */
    }
}
```

In this example, requests to `/bookstore/books/` with a `Range` HTTP request header value of `bytes=0-499` invokes the method with `bytes=0-499` as the value for the `rangeValue` variable.

javax.ws.rs.CookieParam annotation

Cookie parameters are special HTTP headers. While cookies are associated with storing session identification or stateful data that is not accepted as RESTful, cookies can contain stateless information.

If an HTTP cookie is sent, such as `mycustomid=customvalue123`, you can retrieve the value of the `mycustomid` variable using the following example:

```
@javax.ws.rs.Path("/bookstore/books/")
public class BooksCollection {
    @javax.ws.rs.GET
```

```
    public String getBookCollectionInfo(@javax.ws.rs.CookieParam("mycustomid") String mycustomid) {
        /* The cookie value is passed to the mycustomid variable. */
    }
}
```

javax.ws.rs.FormParam annotation

Form parameters are used when submitting a HTML form from a browser with a media type of `application/x-www-form-urlencoded`. The form parameters and values are encoded in the request message body in the form like the following: `firstParameter=firstValue &secondParameter=secondValue`. The javax.ws.rs.FormParam annotation enables easy access to individual form parameter values.

If a form is submitted and the entity value is `firstName=Bob&lastName=Smith`, you can retrieve the values of the form parameters using the following example:

```
@javax.ws.rs.Path("/customer")
public class Custommer {
    @javax.ws.rs.POST
    public String postCustomerInfo(@javax.ws.rs.FormParam("firstName") String firstName, @javax.ws.rs.FormParam("lastName") String lastName) {
        /* firstName would be "Bob" and secondName would be "Smith" */
    }
}
```

Note: You can either use a single unannotated parameter to represent the message body or use multiple FormParam annotated parameters, but not both. Because the FormParam requires the request message body to be read and the message body is represented as a byte stream, the message body cannot be read again. The following code is not valid:

```
@javax.ws.rs.Path("/bookstore/books")
public class BooksCollection {
    @javax.ws.rs.POST
    public String postSpecificBookInfo(@javax.ws.rs.FormParam("bookId") String theBookId, String theRequestEntity) {
        /* This code is invalid.  Can only use FormParam or a request entity parameter like "String theRequestEntity" and not both */
    }
}
```

Choose one of the following ways to define variables to read parameters.

Procedure

- Add a parameter to the resource method by using an appropriate JAX-RS parameter annotation on the method to identify the type of parameter. You can read multiple types of parameters from a request; for example:

```
@javax.ws.rs.Path("/bookstore/books/")
public class BooksCollection {
    @javax.ws.rs.GET
    public String getBookCollectionInfo(@javax.ws.rs.QueryParam("page") int page, @javax.ws.rs.QueryParam("filter") String filter) {
        /* This statement uses the page and filter parameters. */
    }
}
```

  Issuing an HTTP GET request using a web browser or other HTTP client, such as `http://<host_name>:<port>/<context_root>/<servlet_path>/bookstore/books?page=10&filter=FilterValue`, invokes the getBookCollectionInfo() method with the page set to 10 and filter set to `FilterValue`.

- Add the parameter annotation on the fields, JavaBeans properties, and constructor arguments; for example:

```
@javax.ws.rs.Path("/bookstore/books/")
public class BooksCollection {
    @javax.ws.rs.QueryParam("page") int page;

    String filter;

    @javax.ws.rs.QueryParam("filter")
    public void setFilter(String filter) {
        this.filter = filter;
    }

    @javax.ws.rs.GET
    public String getBookCollectionInfo() {
        /* This statement uses the page and filter parameters. */
    }
}
```

Issuing an HTTP GET request using a web browser or other HTTP client, such as `http://<host_name>:<port>/<context_root>/<servlet_path>/bookstore/books?page=10&filter=FilterValue`, also invokes the getBookCollectionInfo() method with the page set to 10 and filter set to `FilterValue`.

Results

Your resource methods are defined so that they can be invoked with appropriate parameter values.

Example

Defining exception mappers for resource exceptions and errors

Java API for RESTful Web Services (JAX-RS) applications can produce exceptions and errors. The default behavior is to use the exception handling functionality of application container such as JavaServer Pages (JSP) error pages. However, you can customize the error handling and send specific responses back when an exception or error occurs.

About this task

JAX-RS resource methods, like any Java method, can throw checked and unchecked exceptions. By default, an unchecked runtime exception or error occurs in the container again. A checked exception is wrapped in a `ServletException` for resources running in the web container. Therefore, a developer can use error handling facilities such as JSP error pages to handle exceptions thrown from a JAX-RS application.

JAX-RS introduced the exception, `javax.ws.rs.WebApplicationException`. A developer can specify a specific error class name or `javax.ws.rs.core.Response` object when creating a `WebApplicationException`. When the `WebApplicationException` is thrown, the information included in the exception by way of a status class name or Response object is used to serialize a response.

If you cannot throw the exception, `WebApplicationException`, in your code and you cannot use the error handling facilities in the web container, but you want to use a custom error response, then you can create a customized JAX-RS `javax.ws.rs.ext.ExceptionMapper` class to map exceptions to HTTP error responses.

The following procedure illustrates how to write a custom `ExceptionMapper` class.

Procedure

1. Create a class that implements the `javax.ws.rs.ext.ExceptionMapper` class and annotate the class with the `javax.ws.rs.ext.Provider` annotation. This step assumes that your JAX-RS resource can throw the exception, `MyCustomException`, in its methods. The following example illustrates a simple `ExceptionMapper` class:

```
import javax.ws.rs.core.Response;
import javax.ws.rs.ext.ExceptionMapper;
import javax.ws.rs.ext.Provider;

@Provider
public class CustomExceptionMapper implements ExceptionMapper<MyCustomException> {

    public Response toResponse(MyCustomException exception) {
        return null;
    }

}
```

2. In the `toResponse(MyCustomException)` method, return a `Response` object that contains the customized error response. The following example illustrates a customized `ExceptionMapper.toResponse(MyCustomException)` method:

```
@Provider
public class CustomExceptionMapper implements ExceptionMapper<MyCustomException> {

    public Response toResponse(MyCustomException exception) {
        return Response.status(500).entity("Unfortunately, the application cannot
```

```
            process your request at this time.").type("text/plain").build();
    }

}
```

You can have additional code where you log an error, inspect the exception thrown, or use more complex logic.

3. Package the compiled custom `ExceptionMapper` class with your web application project. If you rely on the annotation scanning capabilities to find all of your JAX-RS classes in your web application, no additional steps are required. However, if you return all of the relevant JAX-RS resource classes and providers in a JAX-RS application subclass method, then you must also add the custom `ExceptionMapper` class to the returned set. The following example illustrates a preexisting `javax.ws.rs.core.Application` subclass:

```
import java.util.HashSet;
import java.util.Set;

import javax.ws.rs.core.Application;

public class MyApplication extends Application {

    @Override
    public Set<Class<?>> getClasses() {
        Set<Class<?>> classes = new HashSet<Class<?>>();
        classes.add(CustomExceptionMapper.class);
        /* add your additional JAX-RS classes here */
        return classes;
    }
}
```

When exceptions occur in your JAX-RS resource methods, you can customize the HTTP error response so that a user cannot see a stack trace or potentially confidential data. Use an `ExceptionMapper` or the exception handling functionality in the web container to give more helpful responses if the application is not behaving correctly.

Results

You have written a custom `ExceptionMapper` to handle exceptions in your JAX-RS web application.

Deploying JAX-RS web applications

After you assemble your Java API for RESTful Web Services (JAX-RS) web application, you can deploy the web application archive (WAR) package or the enterprise archive (EAR) package onto the application server.

Before you begin

To deploy a JAX-RS web application, you need a WAR package or EAR package that is configured and enabled for RESTful services.

About this task

Every web application must have a context root for the web application to deploy successfully. A context root for each Web module is defined in the application deployment descriptor during application assembly or during application deployment. The context root is combined with the defined servlet mapping from the WAR file to compose the full URL that users type to access the servlet. The context root for each deployed web application must be unique on the server. The context root can also be empty. For instance, if a Web application used a context root of `sample/application/`, the web application request URL begins with `http://<hostname>:<port>/sample/application/`.

The URL pattern of a servlet is appended to the context root of the Web application. For example, if the context root is `sample/application/` and the servlet URL mapping is `rest/api/*`, the base URI for the JAX-RS web application is `http://<hostname>:<port>/sample/application/rest/api`.

Procedure

Deploy the JAX-RS web application WAR package or EAR package onto the application server. Read about installing enterprise application files to learn more about deploying web applications.

Results

The JAX-RS web application is deployed and ready for your business use.

Deployment of a Java API for RESTful Web Services (JAX-RS) web application is successful if you can access the application by typing a Uniform Resource Locator (URL) in a browser or if you can access the application by following a link. If you cannot access your application, follow these steps to eliminate some common errors that can occur during deployment.

Note:

Use the following tips to resolve common errors during deployment of JAX-RS web applications.

An HTTP 404 Not Found error message is sent back to the client in the server response.

To resolve this problem, take the following actions:

- Verify that the root resource classes are annotated with a @javax.ws.rs.Path annotation and that value in the annotation is correct. Root resource classes without a @Path annotation are not registered with the JAX-RS runtime. To learn more, see the defining the URI patterns for resources in RESTful applications information.
- Verify that the root resource class is added to the set of classes returned from the getClasses() method for the subclasses of the javax.ws.rs.core.Application class. Classes not registered in the subclasses of the javax.ws.rs.core.Application class are not recognized by the JAX-RS runtime environment. To learn more, see the defining the URI patterns for resources in RESTful applications information.
- Verify that the web.xml configuration is correct with the expected URL patterns. For additional details, see the configuring the web.xml file for JAX-RS servlets and filters information.
- Verify that the URL that is being used is correct and includes the context root. If you are using a servlet, the servlet URL pattern is a part of the final URL. Using a filter might be more suitable in your web application. For additional details, see the configuring the web.xml file for JAX-RS servlets and filters information.

An HTTP 406 Not Acceptable error message is automatically being sent back to the client in the server response.

To resolve this problem, take the following actions:

- Verify that the Accept HTTP request header in the incoming request is correct. To learn more, see the Implementing content negotiation based on HTTP headers information.
- Verify that the @javax.ws.rs.Produces value on the resource method or resource class is compatible with the incoming Accept HTTP request header. To learn more, see the defining media types for resources in RESTful applications.

An HTTP 415 Unsupported Media Type error message is automatically being sent back to the client in the server response.

To resolve this problem, take the following actions:

- Verify that that the Content-Type HTTP request header in the incoming request is correct and is being sent. To learn more, see the defining media types for resources in RESTful applications.
- Verify that the @javax.ws.rs.Consumes value on the resource method or resource class is compatible with the incoming Content-Type HTTP request header.

An HTTP 204 No Content response status is automatically being sent back to the client in the server response.

To resolve this problem, take the following action:

- If the object returned from a resource method is a null value, then a 204 No Content status response code is sent back from the server runtime automatically.

For information about known problems and their resolution, see the IBM Support page.

IBM Support has documents that can save you time gathering information needed to resolve this problem.

Chapter 36. Administering web services - Security (WS-Security)

The Web Services Security specification defines core facilities for protecting the integrity and confidentiality of a message, and provides mechanisms for associating security-related claims with a message.

Deploying applications that use SAML

After SAML policy sets and bindings have been configured, and SAML tokens created, the SAML token information can be sent from the original login server to other servers using the SAML propagation feature. You can also extract SAML attributes from an existing SAML token and then create additional tokens using the extracted attributes.

About this task

Use the SAML propagation feature of WebSphere Application Server to send SAML token information based on the original login to other servers using a SAML token. Four propagation methods are provided. You can propagate the original SAML token, the SAML token identity and attributes, the WSCredential and WSPrincipal information, or a pre-existing SAML token inserted in the RequestContext.

When SAML is installed on a WebSphere server, you can create SAML attributes using the SAML runtime API. The SAML attributes are added to a CredentialConfig object, which is used to generate a SAML token. The API also provides a function that extracts SAML attributes from an existing SAML token and processes the attributes.

The following topics provide more information about deploying SAML applications.

Propagating SAML tokens

You can use various SAML token propagation methods to include SAML tokens in outbound web services messages.

About this task

A web services client can include two types of tokens in outbound web services messages:

- Original SAML tokens the client received from inbound web services messages.
- New self-issued SAML tokens.

New SAML tokens can be generated using attributes from the original SAML tokens, or using attributes from the WSPrincipal user name in the RunAs Subject. The web services policy configuration determines which SAML tokens will be propagated. You can override the policy configuration by programmatically inserting SAML tokens that you want to propagate into the Axis2 RequestContext object.

Four propagation methods are enabled. This table summarizes the propagation methods and the associated binding options:

Table 272. Propagation methods and associated binding options. Use propagation to include SAML tokens in web services messages.

SAML token propagation method	Binding option	Implementation details
Propagate the original SAML token.	The tokenRequest binding option is set to the value, `propagation`.	Sends the original SAML token from the server where the token was received to other servers using WS-Security.

Table 272. Propagation methods and associated binding options (continued). Use propagation to include SAML tokens in web services messages.

SAML token propagation method	Binding option	Implementation details
Propagate the user security name, unique security name, group IDs, and security realm name.	The tokenRequest binding option is set to the value, `issueByWSCredential`.	Overrides the default system implementation. The self-issued SAML token contains user security name, user unique security name, group IDs, and security realm name that are specified by the WSCredential object in user security context.
Propagate the SAML token identity and attributes.	No binding option is set.	Default system implementation. The server self-generates a new SAML token containing the original SAML attributes, Authentication method, and NameIdentifier or SAML NameID, and sends the new self-generated SAML token to downstream servers using WS-Security. The new SAML token issuer name, issuer signing certificate, and lifetime are determined by the SAML provider configuration properties.
Propagate the WSPrincipal.	The tokenRequest binding option is set to the value, `issueByWSPrincipal`.	Overrides the default system implementation. The self-issued SAML token contains WSPrincipal information in the RunAs subject. The information is stored as NameIdentity or NameID without copying anything from the original SAML token, even if the token exists in the subject.
Programmatically propagate a pre-existing SAML token.	Insert the SAML token that you want to propagate into the RequestContext using the property, `com.ibm.wsspi.wssecurity.core.token.config.WSSConstants.`*`SAMLTOKEN_IN_MESSAGECONTEXT`*.	Overrides all other existing binding options.

Procedure

1. Propagate the original SAML token by setting the tokenRequest binding option to the value, `propagation`, in the bindings.xml file, as shown in the steps. This method sends the original SAML token to other servers using WS-Security. In order for the propagation to succeed, there must be a valid SAML token in the RunAs subject. The server extracts the SAML token from the RunAs subject in the current security context and validates the following conditions. If any of these conditions are invalid, the WS-Security runtime environment does not propagate the SAML token, and the propagation request fails.
   - The SAML token has not expired, and the expiration time is within the time window of the notOnOrAfter value.
   - The ConfirmationMethod setting in the SAML token is the same as the confirmationMethod binding option defined in the token generator configuration.
   - The token ValueType in the SAML token matches the ValueType in the token generator configuration.

   Perform these steps to set the correct value for the tokenRequest binding option. This procedure assumes that a web services client application named JaxWSServicesSamples is deployed, and that the Saml Bearer Client sample binding is attached.
   a. Click **Applications > Application types > WebSphere enterprise Applications > *JaxWSServicesSamples* > Service client policy sets and bindings > *Saml Bearer Client sample* > WS-Security > Authentication and protection**.
   b. Click **gen_saml11token** in the Authentication tokens table.
   c. Click **Callback handler**.
   d. Add the custom property `tokenRequest` and set the property value to `propagation`.
2. To propagate the SAML token identity and attributes using a self-issuing SAML token, modify the outbound tokenGenerator in the bindings.xml file. This method sends the original SAML attributes, NameIdentifier or NameID, and authentication method from the original SAML token to other servers using WS-Security. If there is no SAML token in the subject, the server uses the WSPrincipal, stored as NameIdentifier or NameID, to create a self-issued SAML token. This propagation method is the default system implementation. In this method, the binding option is not set.

The following limitations apply to the bindings.xml file when you are using this propagation method:

- Do not set the tokenRequest binding option in the bindings.xml file.
- Do not set the stsURI binding option in the bindings.xml file, or set the option to this value: `www.websphere.ibm.com/SAML/Issuer/Self`.

3. To propagate the WSPrincipal, modify the bindings.xml file as shown in the steps. Set the tokenRequest binding option to the value, `issueByPrincipal`, in the bindings.xml file. Using this method, the self-issued SAML token is always based on the WSPrincipal even if there is a SAML token in the subject. The new SAML token contains the WSPrincipal user name as the NameId or NameIdentifier. The token does not contain any other attributes in the WSPrincipal or WSCredential objects.

   The following limitation applies to the bindings.xml file when you are using this propagation method:

   - Do not set the stsURI binding option in the bindings.xml file, or set the option to the value, `www.websphere.ibm.com/SAML/Issuer/Self`.

   Perform these steps to set the correct value for the tokenRequest binding option. This procedure assumes that a web services client application named JaxWSServicesSamples is deployed, and that the Saml Bearer Client sample binding is attached.

   a. Click **Applications > Application types > WebSphere enterprise Applications > *JaxWSServicesSamples* > Service client policy sets and bindings > *Saml Bearer Client sample* > WS-Security > Authentication and protection**.

   b. Click **gen_saml11token** in the Authentication tokens table.

   c. Click **Callback handler**.

   d. Add the custom property `tokenRequest` and set the property value to `issueByPrincipal`.

4. To propagate a pre-existing SAML token by inserting SAMLToken in the RequestContext, follow these steps. Use this method to send a SAML token that you created to downstream servers using WS-Security. The propagation is automatically triggered if the WS-Security runtime detects a SAML token in the RequestContext. The pre-existing token overrides any other existing binding options. To use this propagation method, save the existing SAML token in the RequestContext by specifying `com.ibm.wsspi.wssecurity.core.token.config.WSSConstants.`*`SAMLTOKEN_IN_MESSAGECONTEXT`* as the key, as shown in the steps.

   a. Generate a SAML token using the method `SAMLToken samlToken = <token type>`, for example:

```
SAMLToken samlToken = samlFactory.newSAMLToken(cred, reqData, samlIssuerCfg);
```

   b. Save the SAMLToken to the RequestContext, for example:

```
Map requestContext = ((BindingProvider)port).getRequestContext();
   requestContext.put("com.ibm.wsspi.wssecurity.core.token.config.WSSConstants.SAMLTOKEN_IN_MESSAGECONTEXT", samlToken );
```

   This propagation option can co-exist with the other propagation methods, and overrides the other methods. If the SAML token in the RequestContext is expired, or the token expiration time is less than current time plus the cache cushion, the WS-Security runtime environment ignores the SAML token, and uses one of the other three propagation methods that is configured in the bindings.xml file. To avoid using the other three propagation methods, add the following binding option to the custom properties under callback handler in the TokenGenerator configuration: `failOverToTokenRequest = false`.

5. To propagate a user's group memberships, unique security name, and realm name contained in the `com.ibm.websphere.security.cred.WSCredential` object, modify the `bindings.xml` file as shown in the steps. Set the `tokenRequest` binding option to the value, `issueByWSCredential`, in the `bindings.xml` file. Using this method, the self-issued SAML token is always based on the `WSCredential` even if there is a SAML token in the subject.

   The new SAML 1.1 token contains the following assertions:

   - The `NameIdentifier` element contains the `SecurityName` value from `WSCredential` with the `NameQualifier` element set to the realm name from `WSCredential`. The `SecurityName` is obtained by calling the `WSCredential.getSecurityName()` method. The realm name is obtained by calling the `WSCredential.getRealmName()` method.

- All attributes have an AttributeNamespace set to com.ibm.websphere.security.cred.WSCredential as the value.
- The GroupIds attribute contains all group names that a user belongs to. The group names are obtained by calling the WSCredential.getGroupIds() method.
- The UniqueSecurityName attribute contains the unique security name, which is obtained by calling the WSCredential.getUniqueSecurityName() method.
- Optionally, you can assert the realm name from WSCredential by adding the includeRealmName=true custom property in the callback handler.

The new SAML 2.0 token contains the following assertions:

- The NameID element contains the SecurityName value from WSCredential with the NameQualifier element set to the realm name from WSCredential. The SecurityName is obtained by calling the WSCredential.getSecurityName() method. The realm name is obtained by calling the WSCredential.getRealmName() method.
- All attributes have a NameFormat set to com.ibm.websphere.security.cred.WSCredential as the value.
- The GroupIds attribute contains all group names that a user belongs to. The group names are obtained by calling the WSCredential.getGroupIds() method.
- The UniqueSecurityName attribute contains the unique security name, which is obtained by calling the WSCredential.getUniqueSecurityName() method.
- Optionally, you can assert the realm name from WSCredential by adding the includeRealmName=true custom property in the callback handler.

The following limitation applies to the bindings.xml file when you use the propagation method:

- Do not set the stsURI binding option in the bindings.xml file.

Perform these steps to set the correct value for the tokenRequest binding option. This procedure assumes that a Web services client application named JaxWSServicesSamples is deployed, and that the Saml Bearer Client sample binding is attached.

a. Click **Applications > Application types > WebSphere enterprise Applications > *JaxWSServicesSamples* > Service client policy sets and bindings > *Saml Bearer Client sample* > WS-Security > Authentication and protection.**
b. Click **gen_saml11token** in the Authentication tokens table.
c. Click **Callback handler**.
d. Add the tokenRequest custom property and set the property value to issueByWSCredential.

The following example illustrates the NameIdentifier and Attribute statement from a self-issued SAML 1.1 assertion based on WSCredential.

```
<saml:AttributeStatement>
    <saml:Subject>
        <saml:NameIdentifier NameQualifier="ldap.acme.com:9080">uid=alice,dc=acme,dc=com</saml:NameIdentifier>
        <saml:SubjectConfirmation>
            <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:ConfirmationMethod>
        </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Attribute AttributeName="UniqueSecurityName" AttributeNamespace="com.ibm.websphere.security.cred.WSCredential">
        <saml:AttributeValue>uid=alice,dc=acme,dc=com</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="GroupIds" AttributeNamespace="com.ibm.websphere.security.cred.WSCredential">
        <saml:AttributeValue>cn=development,dc=acme,dc=com</saml:AttributeValue>
        <saml:AttributeValue>cn=deployment,dc=acme,dc=com</saml:AttributeValue>
        <saml:AttributeValue>cn=test,dc=acme,dc=com</saml:AttributeValue>
    </saml:Attribute>
</saml:AttributeStatement>
```

The following example illustrates the NameID and Attribute statement from a self-issued SAML 2.0 assertion based on WSCredential.

```
<saml2:AttributeStatement>
    <saml2:Attribute Name="UniqueSecurityName"
           NameFormat="com.ibm.websphere.security.cred.WSCredential">
        <saml2:AttributeValue>uid=alice,dc=acme,dc=com</saml2:AttributeValue>
```

```
    <saml2:Attribute>
    <saml2:Attribute AttributeName="GroupIds"
           NameFormat="com.ibm.websphere.security.cred.WSCredential">
        <saml2:AttributeValue>cn=development,dc=acme,dc=com</saml2:AttributeValue>
        <saml2:AttributeValue>cn=deployment,dc=acme,dc=com</saml2:AttributeValue>
        <saml2:AttributeValue>cn=test,dc=acme,dc=com</saml2:AttributeValue>
    </saml2:Attribute>
</saml2:AttributeStatement>

<saml2:NameID NameQualifier="ldap.acme.com:9060">alice</saml2:NameID>
```

Creating SAML attributes in SAML tokens

Using the SAML runtime API, you can create SAML tokens containing SAML attributes. You can also extract the SAML attributes from an existing SAML token.

About this task

When WebSphere Application Server Version 7.0.0.7 and later is installed, you can create SAML attributes using the SAML token library APIs. The SAML attributes are added to a CredentialConfig object, which is used to generate a SAML token. The API also provides a function that extracts SAML attributes from an existing SAML token and processes the attributes.

To create a SAML token containing SAML attributes, perform the following steps:

Procedure

1. Initialize a com.ibm.wsspi.wssecurity.saml.data.SAMLAttribute object. This creates a SAML attribute based on an address, for example:

```
SAMLAttribute sattribute =
         new SAMLAttribute("urn:oid:2.5.4.20", //Name
       new String[] {" any address"}, //Attribute Values
       null,  /*XML Attributes empty on this example*/
       "urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500", //NameSpace
       "urn:oasis:names:tc:SAML:2.0:attrname-format:uri",  //format
       "Address");
```

2. Use the SAMLTokenFactory to create a CredentialConfig object containing a SAML attribute. This method requires the Java security permisson wssapi.SAMLTokenFactory.newCredentialConfig.
   a. Create a com.ibm.wsspi.wssecurity.saml.config.CredentialConfig object and set a valid principal name.
   b. Create a SAML attribute.
   c. Create a list of SAML attributes and add the SAML attribute to the list.
   d. Add the SAML attribute list to the CredentialConfig object.

   See the following example:

```
SAMLTokenFactory samlFactory =
  SAMLTokenFactory.getInstance("http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0");//samlTokenType

CredentialConfig credentialConfig = samlFactory.newCredentialConfig();
credentialConfig.setRequesterNameID("any name");

SAMLAttribute sattribute =
         new SAMLAttribute("urn:oid:2.5.4.20", //Name
       new String[] {" any address"}, //Attribute Values
       null,  /*XML Attributes empty on this example*/
       "urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500", //NameSpace
       "urn:oasis:names:tc:SAML:2.0:attrname-format:uri",  //format
       "Address");

ArrayList<SAMLAttribute> al = new ArrayList<SAMLAttribute>();
al.add(sattribute);
credentialConfig.setSAMLAttributes(al);
```

3. Specifying the CredentialConfig as a parameter, use the com.ibm.websphere.wssecurity.wssapi.token.SAMLTokenFactory newSAMLToken method to create a SAML token containing the attributes. This step assumes that a RequesterConfig `reqData` object and a ProviderConfig `samlIssuerCfg` object have already been created. For more information on these objects, read about RequesterConfig and ProviderConfig.

a. Obtain an instance of the SAMLTokenFactory.

b. Create a SAML token using the newSAMLToken method from the SAMLTokenFactory, for example:

```
SAMLTokenFactory samlFactory =
  SAMLTokenFactory.getInstance("http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1");

SAMLToken aSamlToken = samlFactory.newSAMLToken(credentialConfig, reqData, samlIssuerCfg);
```

4. Optional: Extract SAML attributes from an existing SAML token. This step is useful to extract the SAML attributes from a received SAML token. You can use this step when a SAML assertion is received and the attributes contained in the assertion need to be processed.

   a. Invoke the getSAMLAttributes() method with the token as a parameter to obtain a list of the SAML attributes in the token. This method requires the Java security permission wssapi.SAMLToken.getSAMLAttributes.

   b. Apply an iterator to the list.

   c. Iterate through the list and perform any additional processing required for your application.

   See the following example:

```
List<SAMLAttribute> aList = aSAMLToken.getSAMLAttributes();
java.util.Iterator<SAMLAttribute> i = aList.iterator();

while(i.hasNext()){

   SAMLAttribute anAttribute = i.next();

   //do something with namespace
   String namespace = anAttribute.getAttributeNamespace();

   //do something with name
   String name = anAttribute.getName();

   //do something with friendly name
   String friendlyName = anAttribute.getFriendlyName();

   //process sring attribute values
   String[] stringAttributeValues = anAttribute.getStringAttributeValue();

   //process XML attribute values
   XMLStructure[] xmlAttributeValues = (XMLStructure[]) anAttribute.getXMLAttributeValue();

   }
```

SAML user attributes

A SAML assertion can contain user attributes relating to the principal of the SAML token. A SAML assertion can contain multiple user attributes.

You can include user attributes in the token to communicate the address of the person who is the SAML assertion principal. This example shows a SAML assertion containing a user attribute:

```
<saml:AttributeStatement>
  <saml:Attribute  xmlns:x500=
      "urn:oasis:names:tc:SAML:2.0:profiles:attribute:X500"
    NameFormat=
      "urn:oasis:names:tc:SAML:2.0:attrname-format:uri"
    Name="urn:oid:2.5.4.20"
    FriendlyName="Address">
          <saml:AttributeValue xsi:type="xs:string">
                      11111 Parker Lane, Austin, Texas, 78758
          </saml:AttributeValue>
  </saml:Attribute>
</saml:AttributeStatement>
```

This table describes the parameters used in the assertion:

Parameter	Description
NameFormat	Specifies how the attribute is interpreted.
Name	Indicates the formal name of the attribute.

Parameter	Description
FriendlyName	Provides a user-friendly name for an attribute when the Name parameter is cryptic.
AttributeValue	The value of the user attribute. The value can be a string, or a complex XML type.

Establishing security context for web services clients using SAML security tokens

WebSphere Application Server supports two policy set caller binding configuration options to establish client security context using SAML security tokens in web services SOAP request messages. The two configuration options are mapping SAML tokens to a user entry in a local user repository and, asserting SAML tokens based on a trust relationship.

Before you begin

This task assumes that you are familiar with WebSphere Application Server SAML technology.

About this task

This task describes setting the WebSphere Application Server policy set caller binding configuration option to establish client security context using SAML security tokens in web services SOAP request messages. You can either map SAML tokens to a user entry in a local user repository or assert SAML tokens based on a trust relationship. The second configuration option does not require accessing the local user repository. Instead, the WS-Security runtime environment populates the client security context entirely using the contents of SAML security tokens. This process is based on a trust relationship to the SAML token issuer. If a SAML tokens specifies the sender-vouches subject confirmation method. the process is based on a trust relationship to the message sender.

Procedure

1. Configure a policy set caller binding and select the SAML token type to represent a web services client request.
   a. Click **WebSphere enterprise applications** > ***application_name*** > **Service provider policy sets and bindings** > ***binding_name*** > **WS-Security** > **Callers**.
   b. Click **New** to create the caller configuration.
   c. Specify a **Name**, such as `caller`.
   d. Enter a value for the **Caller identity local part**. For example, `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0`, which must match the local part of the `CustomToken` element in the attached WS-Security policy.
   e. Click **Apply** and **Save**.
2. Optional: Map SAML security tokens to a user entry in a local user repository. Mapping to a user entry is the default behavior when you configure a caller binding without specifying a configuration option. Alternatively and optionally, you can select this configuration option explicitly using the following steps:
   a. On the caller binding configuration page, add a Callback handler: `com.ibm.websphere.wssecurity.callbackhandler.SAMLIdAssertionCallbackHandler`.
   b. Add a Callback handler custom property, `crossDomainIdAssertion`, and set its value to `false`.
3. Optional: Assert SAML security tokens based on trust relationship.
   a. On the caller binding configuration page, add a Callback handler: `com.ibm.websphere.wssecurity.callbackhandler.SAMLIdAssertionCallbackHandler`.
   b. Add a Callback handler custom property, `crossDomainIdAssertion`, and set its value to `true`.

Note: In WebSphere Application Server Version 7.0 Fix Pack 7 and later releases, the WS-Security runtime environment takes a SAML token Issuer name to represent the foreign security realm name. WS-Security takes the `NameID` element in the case of SAML 2.0 security tokens or the `NameIdentifier` element in the case of SAML 1.1 security tokens to represent user security name. Alternatively, you can explicitly specify which SAML token attribute to use to represent user security name. Moreover, you can also specify which SAML token attribute to use to represent user group membership. Read about SAML assertions across WebSphere Application Server security domains for a detailed discussion of the SAML token assertion trust model and binding configuration.

Version 8 supports propagating select security context data in SAML tokens. You must set a `tokenRequest` custom property with an `issueByWSCredential` property value in the WS-Security binding configuration of the web services client. Read about propagating SAML tokens for a detailed description of this binding option. When the `crossDomainIdAssertion` property is set to `true` in Version 8, WS-Security checks whether a SAML token contains a SAML Attribute `UniqueSecurityName` with a `NameFormat` element with a value of `com.ibm.websphere.security.cred.WSCredential`. If found, WS-Security uses the `NameQualifier` attribute value of the `NameID` element or `NameIdentifier` element to represent the user security realm name. WS-Security also uses the `UniqueSecurityName` attribute value and the `GroupIds` attribute value to represent a unique user name and group membership. This default behavior is different between Version 7 and Version 8 of the product. You can add a `CallbackHandler` property, `IssuerNameForRealm`, and set its value to `true` to configure Version 8 to preserve the Version 7 behavior. Alternatively, you can add a `CallbackHandler` property, `NameQualifierForRealm`, and set its value to `true` to configure Version 8 to always use the `NameQualifier` attribute to represent the user security realm name.

Results

You have configured a web service to establish a client security context using the SAML security token in the web services SOAP request messages.

Example

The following example illustrates the `NameIdentifier` and `Attribute` elements from a self-issued SAML 1.1 assertion based on `WSCredential`:

```
<saml:AttributeStatement>
    <saml:Subject>
        <saml:NameIdentifier NameQualifier="ldap.example.com:9080">uid=alice,dc=example,dc=com</saml:NameIdentifier>
        <saml:SubjectConfirmation>
            <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:ConfirmationMethod>
        </saml:SubjectConfirmation>
    </saml:Subject>
    <saml:Attribute AttributeName="UniqueSecurityName" AttributeNamespace="com.ibm.websphere.security.cred.WSCredential">
        <saml:AttributeValue>uid=alice,dc=example,dc=com</saml:AttributeValue>
    </saml:Attribute>
    <saml:Attribute AttributeName="GroupIds" AttributeNamespace="com.ibm.websphere.security.cred.WSCredential">
        <saml:AttributeValue>cn=development,dc=example,dc=com</saml:AttributeValue>
        <saml:AttributeValue>cn=deployment,dc=example,dc=com</saml:AttributeValue>
        <saml:AttributeValue>cn=test,dc=example,dc=com</saml:AttributeValue>
    </saml:Attribute>
</saml:AttributeStatement>
```

The following example illustrates the `NameID` and `Attribute` elements from a self-issued SAML 2.0 assertion based on `WSCredential`:

```
<saml2:AttributeStatement>
    <saml2:Attribute Name="UniqueSecurityName" NameFormat="com.ibm.websphere.security.cred.WSCredential" />
    <saml2:AttributeValue>uid=alice,dc=example,dc=com</saml2:AttributeValue>
    <saml2:Attribute>
        <saml2:Attribute AttributeName="GroupIds" NameFormat="com.ibm.websphere.security.cred.WSCredential" />
        <saml2:AttributeValue>cn=development,dc=example,dc=com</saml2:AttributeValue>
        <saml2:AttributeValue>cn=deployment,dc=example,dc=com</saml2:AttributeValue>
        <saml2:AttributeValue>cn=test,dc=example,dc=com</saml2:AttributeValue>
```

```
    </saml2:Attribute>
<saml2:AttributeStatement>

<saml2:NameID NameQualifier="ldap.example.com:9060">alice</saml2:NameID>
```

Administering Web Services Security

To secure web services, you must consider a broad set of security requirements, including authentication, authorization, privacy, trust, integrity, confidentiality, secure communications channels, delegation, and auditing across a spectrum of application and business topologies. You can choose to configure Web Services Security for the application level, the server level or the cell level, depending upon your environment and security needs.

Configuring HTTP outbound transport level security with the administrative console

You can configure HTTP outbound transport level security with the administrative console.

Before you begin

This task is one of several ways that you can configure the HTTP outbound transport level security for a web service acting as a client to another web service server. You can also configure the HTTP outbound transport level security with an assembly tool or by using the Java properties. If you do not configure the HTTP outbound transport level security, the web services runtime defers to the Web Services for Java Platform, Enterprise Edition (Java EE) security runtime in the WebSphere product for an effective Secure Sockets Layer (SSL) configuration. If there is no SSL configuration with the Java EE security runtime in the WebSphere product, the Java Secure Socket Extension (JSSE) system properties are used.

About this task

If you choose to configure the HTTP outbound transport level security with the administrative console or an assembly tool, the Web Services Security binding information is modified. You can use the administrative console to configure the web services client security bindings if you have deployed or installed the web services application into WebSphere Application Server. If you have not installed the web services application, you can configure the HTTP SSL configuration with an assembly tool. This task assumes that you have deployed the web services application into the WebSphere product. See the deploying web services applications onto application servers information to learn more about deploying web services.

If you configure the HTTP outbound transport level security using the standard Java properties for JSSE, the properties are configured as system properties. The configuration specified in the binding takes precedence over the Java properties. However, the configurations that are specified by the Java EE security programming model , or that are associated the Dynamic selection , have higher precedence.

See the secure communications using Secure Sockets Layer information to learn more implementing transport level security.

Configure the HTTP outbound transport level security with the following steps provided in this task section.

Procedure

1. Open the administrative console.
2. Click **Applications** > **Enterprise Applications** > *application_instance* > **Manage Modules** > *module_instance* . Under Web Services Security Properties, click **Web Services: Client security bindings**.
3. Under the heading, HTTP SSL Configuration, click **Edit** to access the HTTP SSL configuration panel. Select the **Centrally-managed** radio button so that the system runtime chooses the SSL configuration

that is based on the current context. Select the **Specific to this Web service port** radio button if you want to choose the SSL configuration in the HTTP SSL configuration drop down box.

Results

You have configured the HTTP outbound transport level security for a web service acting as a client to another web service with the administrative console.

HTTP SSL Configuration collection

Use this page to configure transport-level Secure Sockets Layer (SSL) security. You can use this configuration when a web service is a client to another web service.

You can use transport-level security to enable HTTP SSL (or HTTPS). Transport-level security can be enabled or disabled independently from message-level security. Because transport-level security provides minimal security, use message-level security when security is essential to the web service application.

To view this administrative console page, complete the following steps:

1. Click **Applications > Application Types > WebSphere enterprise applications >** ***application_name*** .
2. Click **Manage modules >** ***URI_file_name*** **> Web Services: Client Security Bindings** .
3. Under HTTP SSL Configuration, click **Edit**.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

SSL configuration: Select the **Centrally-managed** radio button so that the system runtime chooses the SSL configuration that is based on the current context. Select the **Specific to this web service port** radio button if you want to choose the SSL configuration in the **HTTP SSL configuration** drop down box.

HTTP SSL configuration: The **HTTP SSL configuration** drop down box lists the SSL configurations used with the HTTP transport for a port. Use this drop down box if you want to select the SSL configuration rather than using the SSL configuration that the runtime automatically selects. To use the drop down box, select the **Specific to the web service port** radio button that is located in the **SSL configuration** field. After you select the radio button, you can click the drop down box to view and select an SSL configuration.

Configuring HTTP outbound transport level security using Java properties

You can configure the HTTP outbound transport level security for a web service using Java properties.

Before you begin

This task is one of three ways that you can configure HTTP outbound transport-level security for a web service that is acting as a client to another web service. You can also configure the HTTP outbound transport level security with the administrative console or an assembly tool. However, you can also use this task to configure the HTTP outbound transport-level security for a web service client.

About this task

If you choose to configure the HTTP outbound transport-level security with the administrative console or an assembly tool, the Web Services Security binding information is modified.

If you configure the HTTP outbound transport-level security using Java properties, the properties are configured as system properties. However, the configuration specified in the binding takes precedence over the Java properties.

You can configure the HTTP outbound transport-level security using WebSphere SSL properties or JSSE SSL properties. However, the WebSphere SSL properties take precedence over the JSSE SSL properties.

Configure the HTTP outbound transport-level security with the following steps provided in this task section.

Procedure

1. Create a property file that includes the following properties:

```
com.ibm.ssl.protocol
com.ibm.ssl.keyStoreType
com.ibm.ssl.keyStore
com.ibm.ssl.keyStorePassword
com.ibm.ssl.trustStoreType
com.ibm.ssl.trustStore
com.ibm.ssl.trustStorePassword
```

2. Set the `com.ibm.webservices.sslConfigURL` Java system property to the absolute path of the created property file. If no WebSphere SSL properties are defined, the JSSE SSL properties are used. Set the JSSE SSL properties as JVM custom properties. See Secure transports with JSSE and JCE programming interfaces for more information about setting the JSSE SSL properties.

Results

You have configured the HTTP outbound transport-level security for a web service acting as a client to another web service.

Configuring HTTP basic authentication for JAX-RPC web services with the administrative console

You can configure HTTP basic authentication for Java API for XML-based RPC (JAX-RPC) web services with the administrative console.

Before you begin

This task is one of three ways that you can configure HTTP basic authentication. You can also configure HTTP basic authentication with an assembly tool or by modifying the HTTP properties programmatically.

If you choose to configure HTTP basic authentication with the administrative console or an assembly tool, the Web Services Security binding information is modified. You can use the administrative console to configure HTTP basic authentication if you have deployed or installed the web services application into WebSphere Application Server. If you have not installed the web services application, then you can configure the security bindings with an assembly tool. This task assumes that you have deployed the web services application into the WebSphere product. To learn more about deploying web services, see the deploying Web services applications onto application servers information.

If you configure HTTP basic authentication programmatically, the properties are configured in the Stub or Call instance. The values set programmatically take precedence over the values defined in the binding.

About this task

The HTTP basic authentication that is discussed in this topic is orthogonal to WS-Security and is distinct from basic authentication that WS-Security supports. WS-Security supports basic authentication token, not HTTP basic authentication.

Configure HTTP basic authentication with the following steps provided in this task section.

Procedure

Open the administrative console.

1. Click **Applications** > **Enterprise Applications** > *application_instance* > **Manage Modules** > *module_instance* > **Web services: Client security bindings**.
2. Click **HTTP Basic Authentication** to access the HTTP basic authentication panel. Enter the values in the HTTP Basic Authentication panel.

Results

You have configured the HTTP basic authentication.

HTTP basic authentication collection

Use this page to specify a user name and password for transport-level basic authentication security for this port. You can use this configuration when a web service is a client to another web service.

You can use transport-level security to enable basic authentication. Transport-level security can be enabled or disabled independently from message-level security. Because transport-level security provides minimal security, use message-level security when security is essential to the web service application.

To view this administrative console page, complete the following steps:

1. Click **Applications > Application Types > WebSphere enterprise applications > *application_name*** .
2. Click **Manage modules > *URI_file_name* > Web Services: Client Security Bindings** .
3. Under HTTP basic authentication, click **Edit**.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

Basic authentication ID:

The user name for the HTTP basic authentication for this port is set in this field.

Use the Basic authentication ID field to specify the user name for the HTTP basic authentication for this port.

Basic authentication password:

The password for the HTTP basic authentication for this port is set in this field.

Use the Basic authentication password field to specify the password for the HTTP basic authentication for this port.

Configuring custom properties to secure web services

You can configure name-value pairs of data, where the name is a property key and the value is a string value that you can use to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available through options in the administrative console.

About this task

The Web Services Security custom properties topic provides the following information about each custom property:

- Provides a detailed description of the property
- States the type of data that is needed to set the property
- Provides a list of possible values
- Lists the default value

Important: Custom properties that you set for the default consumer or default generator bindings take precedence over general custom properties that you set as additional properties. However, custom bindings take precedence over default bindings.

The following steps explain how to set custom properties to secure Web services:

Procedure

- Set custom properties for Java API for XML-based RPC (JAX-RPC) applications. You can set custom properties to secure Web services for JAX-RPC applications in multiple locations within the administrative console. You can set these custom properties for the default consumer, default generator, or both bindings. Also, you can set custom properties as general additional properties. Collectively, the default consumer bindings, the default generator bindings, and the additional properties are referred to as the *default bindings*.
  - **Custom bindings**
    1. Expand **Applications > Application Types**.
    2. Click **WebSphere enterprise applications** > *application_name*.
    3. Under Modules, click **Manage Modules** > *module_name*.
    4. Under Web Services Security Properties, click **Web services: Server security bindings** or **Web services: Client security bindings** > **Edit custom**.
  - **Default consumer bindings**
    1. Expand **Servers > Server types**.
    2. Click **WebSphere applications servers >** *server_name*.
    3. Under Security, click **JAX-WS and JAX-RPC security runtime**.
    4. Under JAX-RPC Default Consumer Bindings, click **Properties**.
  - **Default generator bindings**
    1. Expand **Servers > Server types**.
    2. Click **WebSphere applications servers >** *server_name*.
    3. Under Security, click **JAX-WS and JAX-RPC security runtime**.
    4. Under JAX-RPC Default Generator Bindings, click **Properties**.
  - **Additional properties**
    1. Expand **Servers > Server types**.
    2. Click **WebSphere applications servers >** *server_name*.
    3. Under Security, click **JAX-WS and JAX-RPC security runtime**.
    4. Under Custom properties, click **Custom properties**.

Order of precedence for custom properties with JAX-RPC applications: Custom properties that you set in the WS-Security extension and custom bindings take precedence over custom properties that you set in the default bindings. Custom properties that you set in the WS-Security bindings take precedence over custom properties that you set in the WS-Security extension. Custom properties that you set in the generator or sender and consumer or receiver

bindings take precedence over custom properties that you set in the additional properties.

- Set custom properties for Java API for XML-Based Web Services (JAX-WS) applications. You can set custom properties to secure web services for JAX-WS applications in multiple locations within the administrative console. You can set these custom properties in the custom bindings for an application, in the WS-Security default bindings, or for inbound and outbound messages.
  - **Custom bindings for an application**
    1. Expand **Services > Service clients** or **Services > Service providers**.
    2. Click *service_name* > *binding_name*.
    3. Click **WS-Security**.
    4. Under the Main message security policy bindings heading, click **Custom properties**.
  - **WS-Security default bindings**
    1. Expand **Services > Policy sets** .
    2. Click **General provider policy set bindings** or **General client policy set bindings** > *binding_name* > **WS-Security**.
    3. Under Main Message Security Policy Bindings, click **Custom properties**.
  - **Inbound and outbound custom properties**
    1. Expand **Services > Policy sets**.
    2. Click **Default policy set bindings**.
    3. Under the Policy heading, click **WS-Security**.
    4. Under the Main message security policy bindings heading, click **Custom properties**.

    For more information, see the Inbound and outbound custom properties topic.

  Alternatively, you can set these properties as parameters or inbound binding properties for your JAX-WS application using wsadmin scripting. The following WS-Security policy type property names are used in the setBinding function:
  - application.parameters
  - application.securityinboundbindingconfig.properties
  - application.securityoutboundbindingconfig.properties

  Note: Custom properties for policy set bindings can not be set using the Web Services Security API. The custom properties must be set using the administrative console.

Web services security custom properties

You can configure name-value pairs of data, where the name is a property key and the value is a string value that you can use to set internal system configuration properties. Defining a new property enables you to configure a setting beyond that which is available through options in the administrative console.

Custom properties for Web Services Security can be set in various levels of the application server and for JAX-RPC versus JAX-WS applications. The following list of custom properties provides information on where the custom property is set and how it is used. You can define the following custom properties:

- "Callback handler custom properties for generic security token login modules" on page 3177
- "com.ibm.ws.wssecurity.createSTR" on page 3181
- "com.ibm.wsspi.wssecurity.Caller.assertionLoginConfig" on page 3182
- "com.ibm.wsspi.wssecurity.config.disableWSSIfApplicationSecurityDisabled" on page 3182
- "com.ibm.wsspi.wssecurity.config.gen.checkCacheUsernameTokens" on page 3183
- "com.ibm.wsspi.wssecurity.config.request.setMustUnderstand and com.ibm.wsspi.wssecurity.config.response.forceMustUnderstandEqualsOne" on page 3183

- “com.ibm.wsspi.wssecurity.consumer.timestampRequired” on page 3184
- “com.ibm.wsspi.wssecurity.dsig.inclusiveNamespaces” on page 3185
- “com.ibm.wsspi.wssecurity.token.forwardable” on page 3185
- “com.ibm.wsspi.wssecurity.token.username.addNonce and com.ibm.wsspi.wssecurity.token.username.addTimestamp” on page 3186
- “com.ibm.wsspi.wssecurity.token.username.password.forwardable” on page 3186
- “com.ibm.wsspi.wssecurity.token.username.verifyNonce and com.ibm.wsspi.wssecurity.token.username.verifyTimestamp” on page 3186
- “com.ibm.wsspi.wssecurity.token.UsernameToken.disableUserRegistryCheck” on page 3186
- “com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7” on page 3187

Callback handler custom properties for generic security token login modules

When you configure a generic security token login module, you can specify configuration properties to control how the token is generated or consumed. To configure these custom properties for the callback handler in the administrative console, complete the following steps:

1. Expand **Applications** > **Application Types** and click **WebSphere enterprise applications**.
2. Select an application that contains web services. The application must contain a service provider or a service client.
3. Under the **Web Services Properties** heading, click **Service provider policy sets and bindings** or **Service client policy sets and bindings**.
4. Select a binding. You must have previously attached a policy set and assigned an application-specific binding.
5. Click **WS-Security** in the Policies table.
6. Under the **Main Message Security Policy Bindings** heading, click **Authentication and protection**.
7. Under the **Authentication tokens** heading, click the name of the authentication token.

   Note: You can use the token, which is processed by the generic security token login module, for authentication only. You cannot use the token as a protection token.
8. Under the **Additional Bindings** heading, click **Callback handler**.
9. Under the **Custom Properties** heading, enter the name and value pairs.

The following tables list the custom properties for the callback handler.

Table 273. Callback handler custom properties for both token generator and token consumer bindings.. This table contains the custom property name, its values, and a short description.

Name	Values	Description
`clockSkew`	This custom property does not have a default value.	Use this custom property to specify, in minutes, an adjustment to the times in the self-issued SAML token that the SAMLGenerateLoginModule creates. The clockSkew custom property is set on the Callback handler of the SAML token generator that uses the SAMLGenerateLoginModule class. The value specified for this custom property must be numeric and is specified in minutes. When a value is specified for this custom property, the following time adjustments are made in the self-issued SAML token that the SAMLGenerateLoginModule creates: • The new NotBefore time setting equals the initial NotBefore time setting, minus the amount of time specified for the clockSkew custom property. • The new NotAfter time setting equals the initial NotAfter time setting, plus the amount of time specified for the clockSkew custom property.
`stsURI`	This custom property does not have a default value.	Use this custom property to specify the Security Token Service (STS) address. This custom property is required for the token consumer. However, this custom property is optional for the token generator if the requested token exists in the RunAs Subject and its verification is not required.
`wstrustClientBinding`	This custom property does not have a default value.	Use this custom property to specify the binding name for the WS-Trust client.
`wstrustClientBindingScope`	You can specify an `application` or `domain` value.	Use this custom property to specify the type of bindings that are used for the WS-Trust client. The following conditions apply: • If you specify the `domain` value, general bindings are used. • If you specify the `application` value, custom bindings are used. • If you do not specify a value and application bindings exist, those application bindings are used. • If you do not specify a value and general bindings exist, those general bindings are used. • If neither application or general bindings exist, the default bindings are used. This custom property is optional.
`wstrustClientPolicy`	This custom property does not have a default value.	Use this custom property to specify the policy set name for the WS-Trust client.
`wstrustClientSoapVersion`	You can specify a `1.1` or `1.2` value.	Use this custom property to specify the SOAP message version that the trust client uses to generate the SOAP message. The SOAP message is sent to the Security Token Service (STS). If you do not define this custom property, the generic security token login module uses the SOAP version of the application when it generates the SOAP message for the trust client request. The default value corresponds to the SOAP version that is used by the application client. This custom property is optional.

Table 273. Callback handler custom properties for both token generator and token consumer bindings. (continued). This table contains the custom property name, its values, and a short description.

Name	Values	Description
wstrustClientWSTNamespace	Specify one of the following values: **Trust Version 1.3 (Default)** Specify 1.3 to use Trust Version 1.3 (Default):http://docs.oasis-open.org/ws-sx/ws-trust/200512 **Trust Version 1.2** Specify 1.2 to use Trust Version 1.2:http://schemas.xmlsoap.org/ws/2005/02/trust	Use this custom property to specify which trust client namespace the generic security token login modules uses when it makes the WS-Trust request.
wstrustValidateClientBinding	By default, the value for this custom property is the same value that is specified for the wstrustClientBinding custom property.	Use this custom property to specify the bindings that are used by the WS-Trust Validate request. If you do not specify this custom property, the WS-Trust Validate request uses the same bindings that are used by WS-Trust Issue, which are defined by the wstrustClientBinding custom property.
wstrustValidateClientPolicy	By default, the value for this custom property is the same value that is specified for the wstrustClientPolicy custom property.	Use this custom property to specify the policy sets to use with the WS-Trust Validate request. If you do not specify a value for this custom property, WS-Trust Validate uses the same policy set as WS-Trust Issue, which is defined by the required wstrustClientPolicy custom property.
wstrustIssuer	You can use any string value.	Use this custom property to specify the issuer for the request token. This custom property is optional
wstrustValidateTargetOption	The default value is the WS-Trust Base element extension. You can specify a token value or a base value, which is also the default value.	Use this custom property to specify whether the WS-Trust client passes the validation token to the WS-Trust Security Token Service using the ValidateTarget or the Base element extension. The following conditions apply: • If you do not specify a value for this custom property, the token is wrapped in the Base element extension within the RequestedSecurityToken element. • If you specify the token value, the token is wrapped in the ValidateTarget element within the RequestedSecurityToken element.

Table 274. Callback handler custom properties for token generator bindings only.. This table contains the custom property name, its values, and a short description.

Name	Value	Description
`useRunAsSubject`	You can use a `True` or `False` value. By default, a `True` value is used. This value for this custom property is case sensitive.	Use this custom property to specify whether the generic security token login modules use the token from the RunAs Subject for the outgoing request. By default, the login module uses the validated tokens in the RunAs Subject first. The following conditions apply: • If you set this custom property to a `false` value, the generic security token login module does not use WS-Trust Validate to exchange the token for the outbound request. Instead, it uses WS-Trust Issue to request a token. • If you do not specify this custom property, the generic security token login module attempts to use a token from the RunAs Subject and WS-Trust Validate to exchange the token. • If a token does not exist in the RunAs Subject, the generic security token login module uses WS-Trust Issue and is protected by the trust client policy sets.
`useRunAsSubjectOnly`	You can use a `True` or `False` value. By default, a `False` value is used. This value for this custom property is case sensitive.	Use this custom property to disable or enable WS-Trust Issue in the generic security token login module. If you set this custom property to a `true` value, the generic security token login module uses the token from the RunAs Subject and WS-Trust Validate to exchange the tokens. The generic security token login module does not use WS-Trust Issue to request a token even if WS-Trust Validate fails or it does not find a matching token in the RunAs Subject.
`useToken`	You can use any string value of the ValueType value for the security token.	When you use a security token in a RunAs Subject to validate and exchange tokens for an outbound request, you can use this custom property to specify which token ValueType value in the RunAs Subject to validate and exchange for the requested token. For example, you might have a token with a ValueType value of *Token_1* in the RunAs Subject. However, the ValueType value of *Token_2* is the required token. You can set this custom property to *Token_1* . If you do not define this custom property, the validation token is the token from the RunAs Subject that has the same ValueType value as the required token. This custom property is optional.

Table 274. Callback handler custom properties for token generator bindings only. (continued). This table contains the custom property name, its values, and a short description.

Name	Value	Description
validateUseToken	You can use a True or False value. By default, a True value is used. This value for this custom property is case sensitive.	Use this custom property to specify whether the token generator uses WS-Trust Validate to validate the token from the RunAs Subject. By default, the generic security token login module validates a token from the RunAs Subject against the Security Token Service (STS) before sending the token in the SOAP message to the service provider. If you set this custom property value to false and the generic security token login module finds a matching token from the RunAs Subject, the login module does not invoke WS-Trust Validate to validate the matching token. Instead, it sends the matching token to the downstream service provider without validation.
wstrustIncludeTokenType	You can use a True or False value. By default, a True value is used. This value for this custom property is case sensitive.	Use this custom property to specify whether the WS-Trust RequestedSecurityToken token includes the requested token ValueType value. If you do not specify this custom property, the generic security token login modules includes the requested token type in the WS-Trust RequestedSecurityToken token. This custom property is optional.

Table 275. Callback handler custom properties for token consumer bindings only.. This table contains the custom property name, its values, and a short description.

Name	Value	Description
exchangedTokenType	The valid value for this custom property is the string ValueType value for the token that is supported by the system default login modules.	Use this custom property to specify the new token with the defined ValueType value, which the trust service must return after successful validation. If you do not specify a value for the custom property, the generic security token login module accepts whichever token the trust service returns. This custom property is optional.

com.ibm.ws.wssecurity.createSTR

The com.ibm.ws.wssecurity.createSTR property creates a security token reference to the security token in the SOAP security header when you specify a True value.

You can set this property to True, the com.ibm.ws.wssecurity.createSTR property creates a security token reference to the security token in the SOAP security header. Set this custom property to True when the following conditions exist:

- The referencing mechanism for the token signature is the STR Dereference Transform, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform
- The SignedParts element for the WS-Security policy contains an XPath value that represents the SecurityTokenReference.

Data type	String
Values	True, False
Default	False

The value for this property is case-insensitive.

com.ibm.wsspi.wssecurity.Caller.assertionLoginConfig

The com.ibm.wsspi.wssecurity.Caller.assertionLoginConfig property, which is configured on the caller part, specifies the name of the JAAS login configuration that is used by Web Services Security to obtain WebSphere Application Server authorization credentials. You must configure this property using an assembly tool such as the Rational Application Developer. For more information, see the "Configuring the caller in consumer security constraints" topic for Rational Application Developer. Within this topic, this custom property is set when you configure identity assertion.

Use this property with WS-Security V1.0 JAX-RPC applications only.

Data type	String
Default	system.DEFAULT

com.ibm.wsspi.wssecurity.config.disableWSSIfApplicationSecurityDisabled

When you set the com.ibm.wsspi.wssecurity.config.disableWSSIfApplicationSecurityDisabled custom property to `true`, Web Services Security does not enforce the configured WS-Security constraints if application security is disabled on the application server. You can use this custom property to debug services in a non-secure environment without needing to remove security constraints from web services applications.

Note: Use this custom property for diagnosis purposes only. Do not use it in a production environment.

Data type	String
Values	true, false
Defaultfalse	false

You can set this custom property as an inbound custom property or an inbound and outbound custom property for the policy set bindings. Complete the following steps in the administrative console to set the custom property:

1. Expand **Services** > **Policy sets**.
2. Click **General provider policy set bindings** or **General client policy set bindings**.
3. Click the *binding_name*.
4. Under the Policy heading, click **WS-Security** > **Custom properties**.

You can also set this custom property as a parameter or as an inbound binding property on your application using wsadmin tooling. The following WS-Security policy-type property names are used in setBinding:

- application.parameters
- application.securityinboundbinding config.properties

com.ibm.wsspi.wssecurity.config.gen.checkCacheUsernameTokens

The com.ibm.wsspi.wssecurity.config.gen.checkCacheUsernameTokens custom property specifies whether to cache UsernameTokens all of the time, which is the default behavior, or cache them as determined by a set of rules. You can configure this custom property for the token generator or as an additional property.

When the com.ibm.wsspi.wssecurity.config.gen.checkCacheUsernameTokens custom property is set to false, UsernameTokens are always cached on client threads. When you set this custom property to true, the web services security run time determines whether UsernameTokens are cached based on the following rules:

- Never cache UsernameTokens if the application is running on an application server.
- Cache UsernameTokens if the token generator for the UsernameToken has the following callback handler configured: com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler.

This custom property applies to the JAX-RPC run time only. Use an assembly tool, such as Rational Application Developer, to set the custom property within the encrypted message part bindings.

Data type	String
Values	true, false
Default	false

com.ibm.wsspi.wssecurity.config.request.setMustUnderstand and com.ibm.wsspi.wssecurity.config.response.forceMustUnderstandEqualsOne

In WebSphere Application Server prior to Version 6.1x, the mustUnderstand=1 attribute in the `<wsse:Security>` tag in the SOAP header on the request from the web services client was hardcoded. It was not possible to configure the mustUnderstand attribute in the SOAP Web Services Security header. In an update to the product, an administrator can configure the attribute using outbound generator custom properties.

You can configure the following outbound generator custom properties for Web Services Security:

com.ibm.wsspi.wssecurity.config.request.setMustUnderstand custom property

The `com.ibm.wsspi.wssecurity.config.request.setMustUnderstand` custom property specifies the mustUnderstand setting in outbound consumer requests. If the value of the property is set to zero (0), no, or false, then the mustUnderstand attribute is not set in the WS-Security header within outbound consumer requests.

Data type	String
Value	Zero (0), no, false
Default	true

In SOAP messages, the default value for the mustUnderstand attribute is zero (0). According to the SOAP specification, if the intended value for the attribute is zero, then the attribute must not be present in the message.

com.ibm.wsspi.wssecurity.config.response.forceMustUnderstandEqualsOne custom property

The `com.ibm.wsspi.wssecurity.config.response.forceMustUnderstandEqualsOne` custom property specifies that the provider should always respond with a `mustUnderstand="1"` attribute in the SOAP security header. If the value is set to one (1), yes, or true, the provider responds with the `mustUnderstand="1"` attribute in the WS-Security header. The default value of the attribute is false.

Data type	String
Value	One (1), yes, or true
Default	false

By default, the response contains the same mustUnderstand attribute as the request. For example, if the inbound request has mustUnderstand="1", the response also includes mustUnderstand="1". If the request does not have a mustUnderstand attribute, the response does not include a mustUnderstand attribute.

For JAX-RPC applications, you can specify both properties in the following locations within the administrative console:

- Click **Servers** > **Server Types** > **WebSphere application servers** > ***server name***. Under Security, click **JAX-WS and JAX-RPC security runtime**. Under JAX-RPC Default Generator Bindings, click **Properties**.
- Click **Servers** > **Server Types** > **WebSphere application servers** > ***server name*** . Under Security, click **JAX-WS and JAX-RPC security runtime**. Under Custom properties, click **Custom properties**.

If you are using an assembly tool with a JAX-RPC WS-Security version 1.0 application, you can set the `com.ibm.wsspi.wssecurity.config.request.setMustUnderstand` custom property on the security request generator extension or binding. You can set the `com.ibm.wsspi.wssecurity.config.response.forceMustUnderstandEqualsOne` custom property on the response generator extension or binding. A setting in the binding takes precedence over a setting in the extension.

If using an assembly tool with a JAX-RPC WS-Security specification draft 13–level application, you can set the `com.ibm.wsspi.wssecurity.config.request.setMustUnderstand` custom property as a parameter on the port qualified name binding. You can set the `com.ibm.wsspi.wssecurity.config.response.forceMustUnderstandEqualsOne` custom property as a parameter on the port component binding.

If using a JAX-WS application , you can set the custom properties as outbound binding properties or parameters on the application using wsadmin scripts. The following property names are used:

- `application.parameters`
- `application.securityoutboundbindingconfig.properties`

However, properties values in the `application.securityoutboundbindingconfig.properties` properties take precedence over properties in application parameters. The follow example shows how to use Jython wsadmin commands to obtain the ID of the policy set attachment for a consumer, then set the `com.ibm.wsspi.wssecurity.config.request.setMustUnderstand` property to false in the outbound binding configuration:

```
AdminTask.getPolicySetAttachments([-applicationName
HelloSvcClientEAR -attachmentType client])

AdminTask.setBinding([-policyType WSSecurity -bindingLocation "[
[application HelloSvcClientEAR] [attachmentId 1490] ]"
-attributes "[[application.securityoutboundbindingconfig.properties_999.name
com.ibm.wsspi.wssecurity.config.request.setMustUnderstand]
[application.securityoutboundbindingconfig.properties_999.value
false]]" -attachmentType client])
```

com.ibm.wsspi.wssecurity.consumer.timestampRequired

The com.ibm.wsspi.wssecurity.consumer.timestampRequired property specifies whether Timestamp is not expected in the security header for the response when the **Include timestamp in security header** setting is selected for the WS-Security policy.

The JAX-WS WS-Security runtime is updated to comply with the OASIS WS-SecurityPolicy 1.2 specification Timestamp Required requirement. If you want to configure an application to not require an inbound time stamp when an outbound time stamp is configured you can add the com.ibm.wsspi.wssecurity.consumer.timestampRequired custom property to your Web Services Security

settings and set that property to `false`. When this property is set to `false`, even if the **Include timestamp in security header** is selected as a setting for the WS-Security policy, a Timestamp is not expected in the security header for a response.

The default value for this property is true.

gotcha: On the custom properties panel, you can set this property as either an inbound or an inbound/outbound custom property. It is not valid as an outbound custom property.

Data type	Boolean
Default	true

com.ibm.wsspi.wssecurity.dsig.inclusiveNamespaces

This custom property, which applies to both the JAX-RPC and JAX-WS applications, specifies whether to disable the inclusive namespace prefix list for XML digital signatures. WebSphere Application Server, by default, includes the prefix in the digital signature for Web Services Security. You can set this custom property to false if you do not want inclusive namespaces set as an element. Some implementations of Web Services Security cannot handle this prefix list. If you experience a signature validation failure when a signed SOAP message is sent and you are using another vendor in your environment, check with your service provider for a possible fix to their implementation before you disable this property.

For JAX-RPC applications, you can set the custom property in the administrative console in the signing information or as a web services security custom property in additional properties or in the default or custom generator bindings. For more information, see the additional properties and generator sections of the Configuring custom properties to secure web services topic. To add the custom property to the signing information, complete the following steps:

1. Click **Applications** > **Enterprise Applications** > ***application_name***.
2. Click **Manage Modules** > ***module_name***.
3. Under Web Services Security Properties, click **Web services: Client security bindings** or **Web services: Server security bindings**.
4. Under Request generator (sender) binding or Response generator (sender) binding, click **Edit custom**.
5. Under Required properties, click **Signing information** > ***signing_information_name*** > **Properties**.
6. Specify the custom property and its value.

For JAX-WS applications, you can configure this custom property in the outbound signing information. To configure the custom property, complete the following steps:

1. Click **Services** > **Service clients** or **Services** > **Service providers**.
2. Click the ***service_name*** > ***binding_name***.
3. Under Policy, click **WS-Security**
4. Under Message Security Policy Bindings, click **Authentication and protection**
5. Under either Request message signature and encryption protection or Response message signature and encryption protection, click the *signature_message_part_reference*. When you click the *signature_message_part_reference* name, you are accessing the configuration for the signed message part binding.
6. Specify the custom property and its value.

com.ibm.wsspi.wssecurity.token.forwardable

When configuring SecurityToken consumer bindings for the JAX-WS programming model, use this custom property to specify whether the receiving token is propagated to other servers. If you specify a value of `true` for this property, you enable this token for propagating to other servers. If you specify a value of

`false` for this property, the token is not propagated to other servers. The default value is `true`, and the value is not case sensitive.

com.ibm.wsspi.wssecurity.token.username.addNonce and com.ibm.wsspi.wssecurity.token.username.addTimestamp

When configuring a username token for the JAX-WS programming model, to protect against replay attacks it is strongly recommended that you add custom properties, com.ibm.wsspi.wssecurity.token.username.addNonce and com.ibm.wsspi.wssecurity.token.username.addTimestamp, to the callback handler configuration for token generation. These custom properties enable and verify the nonce and timestamp for message authentication. The value of the properties must be set to `true`.

com.ibm.wsspi.wssecurity.token.username.password.forwardable

When configuring UsernameToken consumer bindings for the JAX-WS programming model, use this custom property to specify whether the password is propagated along with the UsernameToken to other servers during UsernameToken propagation. If you specify a value of `true` for this property, the password is preserved during propagation. If you specify a value of `false` for this property,the password must be removed prior to UsernameToken propagation. The default value is `true`, and the value is not case sensitive.

com.ibm.wsspi.wssecurity.token.username.verifyNonce and com.ibm.wsspi.wssecurity.token.username.verifyTimestamp

When configuring a username token for the JAX-WS programming model, to protect against replay attacks it is strongly recommended that you add custom properties, com.ibm.wsspi.wssecurity.token.username.verifyNonce and com.ibm.wsspi.wssecurity.token.username.verifyTimestamp, to the callback handler configuration for the token consumer. These custom properties enable and verify the nonce and timestamp for message authentication. The value of the properties must be set to `true`.

com.ibm.wsspi.wssecurity.token.UsernameToken.disableUserRegistryCheck

This propery allows the user registry check to be skipped for identity tokens. This means that the user name associated with the identity token in an identity assertion scenario can pass through the UNTConsumeLoginModule without generating a registry error. Typically an identity token must not contain a password, and there might, or might not be a trust token. For example there might be a blind trust.

This property does not affect any UsernameToken that contains a password. If you need to bypass the registry check for a UsernameToken that contains a password, the UNTConsumeLoginModule that is provided with the product cannot be used. The following WS-Security custom property is added to the UNTConsumeLoginModule to allow the user registry check to be skipped for identity tokens:

When the property is set to true, the UNTConsumeLoginModule does not validate the inbound UsernameToken if, and only if, the UsernameToken does not contain a password.

Valid values for this property are true and false. The default value is false.

To configure this property, in the administrative console:

1. Expand **Services** > **Policy sets**.
2. Click **General provider policy set bindings** or **General client policy set bindings**.
3. Click the binding name.
4. Under the Policy heading, click **WS-Security** > **Authentication and Protection** > ***tokenName*** > **Callback Handler**.

5. Add this property and its value in the Custom Properties **Name** and **Value** fields.

com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7

Web services security supports both LTPA (Version 1) and LTPA Version 2 (LTPA2) tokens. The LTPA2 token, which is more secure than Version 1, is supported by the JAX-WS run time only. You can set the **Enforce token version** interoperability option on the token generator to determine whether an LTPA (Version 1) or an LTPA2 token is retrieved when a request message is received. However, if you want to force the run time to use LTPA (Version 1) tokens only, you can set the com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7 custom property to true

To enable this custom property, complete the following steps in the administrative console:

1. Locate the binding that you want to configure.
2. Click the WS-Security policy in the Policies table.
3. Click the Authentication and protection link in the security policy bindings section.
4. Click the token generator that you want to configure.
5. Specify com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7 to true in the Custom properties section.

The following table explains how combinations of this custom property and the **Enforce token version** interoperability option affect the runtime.

Table 276. LTPA interoperability. Table of the com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7 custom property and **Enforce token version** *interoperability option values.*

com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7 custom property value	**Enforce token version value**	**Result**
false	Disabled	The run time can use both LTPA (Version 1) and LTPA2 tokens.
not specified, which implies a false value	Disabled	The run time can use both LTPA (Version 1) and LTPA2 tokens.
true	Disabled	The run time can use LTPA (Version 1) tokens only.
true	Enabled	The run time can use LTPA (Version 1) tokens only.

For more information, see the documentation about enabling or disabling single sign-on interoperability mode for the LTPA token.

Administering message-level security for JAX-WS web services

Web Services Security standards and profiles describe how to provide security and protection for SOAP messages that are exchanged in a web services environment. Using JAX-WS, development of web services and clients is simplified with greater platform independence for Java applications through the use of dynamic proxies and Java annotations.

Auditing the Web Services Security runtime

Security auditing provides tracking and archiving of auditable events for the web services runtime operations. When security auditing is enabled for web services, the event generator utility collects and logs signing, encryption, security, authentication, and delegation events in audit event records. You can analyze the audit event records to identify possible security breaches or potential weaknesses in the security configuration of your environment.

The audit security subsystem must be enabled for WebSphere Application Server before the event generator can collect auditing records for the Web Services Security runtime. The recording of auditable security events is achieved by enabled the security auditing subsystem. For more information about enabling security auditing, read the topic "Enabling the security auditing subsystem."

Web Services Security auditing is enabled for the JAX-WS runtime only. Several auditing events can occur when a SOAP message is received. Auditing data is collected during various Web Services Security runtime operations, such as validating the digital signature of a SOAP message, decrypting the message, or checking the message security header. The auditing data is stored and managed by the event generator, and stored in audit logs for later analysis.

Auditable events for web services include:

Signing

As the digital signature in each SOAP message part is validated, a SECURITY_SIGNING event is sent to the event generator, along with an outcome, which can be SUCCESS or ERROR. Reason codes, either VALID_SIGNATURE or INVALID_SIGNATURE, are also sent. The integrity of the message is audited, also with a SECURITY_SIGNING event, with the outcome of SUCCESS or DENIED. Reason codes are INTEGRITY or INTEGRITY_BAD. The following table summarizes the signing events:

Table 277. Signing audit events. Use the signing audit event records to identify possible security breaches or weaknesses in the security configuration.

Event type	Possible outcomes	Reason codes
SECURITY_SIGNING (digital signature)	SUCCESS ERROR	VALID_SIGNATURE INVALID_SIGNATURE
SECURITY_SIGNING (integrity)	SUCCESS DENIED	INTEGRITY INTEGRITY_BAD

If the SECURITY_SIGNING event outcome is DENIED, and the reason code is INTEGRITY_BAD, this means that at least one message part, which must be signed by the security policy, failed signature validation. If the SECURITY_SIGNING event outcome is ERROR, and the reason code is INVALID_SIGNATURE, this means that the digital signature validation failed. This could lead to an INTEGRITY_BAD reason code in the audit record, depending on whether the digital signature is required by the security policy.

Encryption

Encryption auditing is performed in two parts: first, the encrypted parts of the SOAP message are processed, then the confidentiality of the message is audited. To audit each encrypted part of the message, a SECURITY_ENCRYPTION event is sent. An event record is created only if the outcome of the event is ERROR, meaning that an exception is encountered. The reason code is DECRYPTION_ERROR. Next, the confidentiality of the message is checked with another SECURITY_ENCRYPTION event, which has possible outcomes of SUCCESS or DENIED. Reason codes for this event are CONFIDENTIALITY or CONFIDENTIALITY_BAD. The following table summarizes the encryption events:

Table 278. Encryption audit events. Use the encryption audit event records to identify possible security breaches or weaknesses in the security configuration.

Event type	Possible outcomes	Reason codes
SECURITY_ENCRYPTION (encrypted message parts)	ERROR	DECRYPTION_ERROR
SECURITY_ENCRYPTION (confidentiality)	SUCCESS DENIED	CONFIDENTIALITY CONFIDENTIALITY_BAD

If the SECURITY_ENCRYPTION event outcome is DENIED and the reason code is CONFIDENTIALITY_BAD, this means that at least one message part, which is required to be encrypted, is not correctly encrypted. A DECRYPTION_ERROR in the audit record could lead to a CONFIDENTIALITY_BAD reason code, depending on whether message encryption is required.

Time stamp

For each SOAP message, the time stamp is audited when a SECURITY_RESOURCE_ACCESS event is sent to the event generator. The possible outcomes of this event are SUCCESS or DENIED. Reason codes are TIMESTAMP or TIMESTAMP_BAD. The following table summarizes the time stamp events:

Table 279. Time stamp audit event. Use the time stamp audit event record to identify possible security breaches or weaknesses in the security configuration.

Event type	Possible outcome	Reason code
SECURITY_RESOURCE_ACCESS	SUCCESS DENIED	TIMESTAMP TIMESTAMP_BAD

Security header

The security header is audited to make sure it is not missing from the SOAP message. A SECURITY_RESOURCE_ACCESS event is sent, and if the header is missing, the outcome is DENIED, with the reason code SECURITY_HEADER_MISSING. The following table summarizes the security header event:

Table 280. Security header audit event. Use the security audit event record to identify possible security breaches or weaknesses in the security configuration.

Event type	Possible outcome	Reason code
SECURITY_RESOURCE_ACCESS	DENIED	SECURITY_HEADER_MISSING

Authentication

The security token used to authenticate a message is audited to determine if authentication is successful, and information about the authentication type, provider name and provider status are saved in the audit event record. The token ID is also recorded, along with information that is specific to the token type, such as username or keystore. Sensitive information such as the token itself, or the token password, is not recorded in the audit event record. Information recorded for each token type is described in the following table:

Table 281. Token information recorded in audit event record. Use the authentication audit event records to identify possible security breaches or weaknesses in the security configuration.

Token type	Recorded information
Username token	Username Password (null or not-null) Token ID
LTPA	Principal Expiration Token ID
LTPAPropagate	Principal Expiration Token ID
SecureConversation	UUID Instance UUID Token ID
DerivedKey	Reference ID Token ID
Kerberos	Principal SPN pSHA1 Token ID
X509	Certificate Subject Issuer Keystore Token ID TrustAny
PKP Path (public key infrastructure)	Certificate Subject Issuer Keystore Token ID TrustAny

Table 281. Token information recorded in audit event record (continued). Use the authentication audit event records to identify possible security breaches or weaknesses in the security configuration.

Token type	Recorded information
PKCS7 (Public Key Cryptography Standards)	Certificate Subject Issuer Keystore Token ID TrustAny
SAML token	Principal SAML Token Issuer Name Confirmation Method

Message authentication is audited using a SECURITY_AUTHN event. Possible outcomes of the event are SUCCESS, DENIED or FAILURE. Reason codes are AUTHN_SUCCESS, AUTHN_LOGIN_EXCEPTION or AUTHN_PRIVILEDGE_ACTION_EXCEPTION. The following table summarizes the authentication events:

Table 282. Authentication audit event. Use the authentication audit event records to identify possible security breaches or weaknesses in the security configuration.

Event type	Possible outcomes	Reason codes
SECURITY_AUTHN	SUCCESS DENIED FAILURE	AUTHN_SUCCESS AUTHN_LOGIN_EXCEPTION AUTHN_PRIVILEDGE_ACTION_EXCEPTION

Delegation

Authentication of a SOAP message can be delegated, and the delegation function is audited using a SECURITY_AUTHN_DELEGATION event. This event is sent when the client identity is propogated or when delegation involves the use of a special identity. In the Web Services Security runtime, auditing events are recorded only when the delegation type is set to Identity Assertion. Possible outcomes are SUCCESS or DENIED, with reason codes AUTHN_SUCCESS or AUTHN_DENIED. Additional information, such as delegation type, role name and identity name, is collected and stored in the audit event record. The following table summarizes the delegation events:

Table 283. Delegation audit event. Use the delegation audit event record to identify possible security breaches or weaknesses in the security configuration.

Event type	Possible outcomes	Reason codes
SECURITY_AUTHN_DELEGATION	SUCCESS DENIED	AUTHN_SUCCESS AUTHN_DENIED

Validation

As part of the generic security token login module support, a Security Token Service can validate a token using a WS-Trust Validate request. This validation operation can return a token in exchange for a token that is being validated. The information that is exchanged is located in the documentation on auditing for a generic security token. The following table shows the token exchange information that is recorded.

Table 284. Token exchange. Use the token exchange information to trace the token exchange process.

Event	Description
TokenSentForExchangeId	This information identifies the token that is sent for validation and is exchanged.
TokenSentForExchangeType	This information identifies the type of the token that is sent for validation and is exchanged.
ExchangedTokenId	This information identifies the token that is received in exchange during the validation request.
ExchangedTokenType	This information identifies the type of the token that is received in exchange during the validation request.

If the token is validated without a token exchange, only the Token ID is recorded.

Securing web services using policy sets

Policy sets are assertions about how services are defined. They are used to simplify the quality of service configuration for web services.

About this task

Policy sets combine configuration settings, including those for transport and message level configuration, such as WS-Addressing, WS-ReliableMessaging, and WS-Security. There are two main types of policy sets; application policy sets and system policy sets. Application policy sets are used for business-related assertions. These assertions are related to the business operations that are defined in the Web Services Description Language (WSDL) file. System policy sets, on the other hand, are used for non-business-related system messages. These messages are not related to the business operations that are defined in the WSDL, but instead refer to messages that are defined in other specifications which apply qualities of service (QoS). Such QoS are the request security token (RST) messages that are defined in WS-Trust, or create sequence messages that are defined in WS-Reliable Messaging metadata exchange messages of the WS-MetadataExchange.

Note: You can use policy sets only with Java™ API for XML-Based Web Services (JAX-WS) applications. You cannot use policy sets with Java API for XML-based RPC (JAX-RPC) applications.

Policies are defined based on a quality of service. Policy definition is typically based on WS-Policy standard language, for example, the WS-Security policy is based on the current WS-SecurityPolicy from the Organization for the Advancement of Structured Information Standards (OASIS) standards.

Policy sets do not include environment or platform-specific information, such as keys for signing, keystore information, or persistent store information. This type of information is defined in the binding. A policy set attachment defines how a policy set is attached to service resources and bindings. The attachment definition is outside the policy set definition and is defined as meta-data associated with application data.

To secure JAX-WS web services with message-level security using policy sets, follow these steps:

Procedure

1. Select, create, or copy and modify a policy set to specify the message-level protection required. The policy specifies what protection will be applied, for example, what message parts to sign or encrypt and the token types and algorithms to use.
   - Select one of the web services policy sets.
   - Create, copy, modify, import, export or delete a policy set. For more information, read about managing policy sets using the administrative console
2. Attach the policy set to the application.
3. Create or select the policy set bindings to be used. The bindings are then attached to the application along with the policy set. The bindings used can either be general bindings that can be shared among applications or application specific bindings. For more information, read about defining and managing policy set bindings.
4. If WS-SecureConversation is being used, specify the trust service system policy sets and bindings on the application server.

Example: Configuring the message-level WS-Security policy set and bindings:

This example shows how to configure the message-level WS-Security policy set and bindings to send a Username token in a JAX-WS request, and to encrypt the Username token using asymmetric encryption.

Before you begin

Make a copy of the **Username WSSecurity** default policy set and give it a unique name. This example illustrates how to modify a copy of the default policy set. For more information, read about copying default policy set and bindings settings.

About this task

By default, the Username WSSecurity policy set signs the WS-Addressing headers and body in the request and the response, and encrypts the body and signature in the request and the response. However, in this example, the goal is to encrypt only the Username token in the request from the client to the service, but not to encrypt any part of the response from the service to the client. In addition, no part of the request or the response will be signed. Therefore, the policy set must be modified to remove several message protection parts. You must also configure the client and server bindings.

First, configure the policy set by modifying your copy of the Username WSSecurity default policy set.

Procedure

1. From the administrative console, click **Services > Policy sets > Application policy sets > *policy_set_name***. In the Policy set settings panel, you can specify information about the policy set, such as the description.
2. Remove the following message protection parts: request:app_signparts, response:app_signparts and response:app_encparts.
   a. Click **Application policy sets > *policy_set_name* > WS-Security > Main policy > Response message part protection**.
   b. Click on **app_encparts** in the Encrypted parts box, then click the **Delete** button.
   c. Click on **app_signparts** in the Signed parts box, then click the **Delete** button.
   d. Click **Application policy sets > *policy_set_name* > WS-Security > Main policy > Request message part protection**.
   e. Click on **app_signparts** in the Signed parts box, then click the **Delete** button.
3. Update the protection part specified for request:app_encparts. By default, this message protection part encrypts the body and signature elements, and must be modified to encrypt the Username token.
   a. Click **Application policy sets > *policy_set_name* > WS-Security > Main policy > Request message part protection > Encrypted part - app_encparts > Edit**.
   b. Delete the existing elements in the Elements in part panel, then add two XPath expressions for encrypting the Username token.

      Expression 1:

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/'
        and local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/'
        and local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
        and local-name()='Security']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
        and local-name()='UsernameToken']
```

      Expression 2:

```
/*[namespace-uri()='http://www.w3.org/2003/05/soap-envelope'
        and local-name()='Envelope']/*[namespace-uri()='http://www.w3.org/2003/05/soap-envelope'
        and local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
        and local-name()='Security']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
        and local-name()='UsernameToken']
```

What to do next

The second part of the process is to configure the client and server bindings.

gotcha: When working with the client binding, be careful when choosing the protection token that you want to edit. The names of the tokens are represented from the perspective of the protector. For instance, on a client, even though the word recipient is in its name, the

AsymmetricBindingRecipientEncryptionToken0 is a generator. The best way to make sure that you are choosing the correct token is to look at the Usage column in the Protection tokens table.

1. Configure the client binding, as follows:
   a. Attach the policy to a service resource and create a new binding for that resource that includes the WSSecurity policy.
   b. Click on **WSSecurity** in the new binding to display the main WSSecurity binding panel. For example, click **Enterprise applications > WSSampleServiceSei > Service client policy sets and bindings > *binding_name* > WS-Security**.
   c. Click **Authentication and protection**.
   d. Under Protection tokens, click **AsymmetricBindingRecipientEncryptionToken0** (the Asymmetric encryption generator).
   e. Click **Apply**.
   f. Click **Callback handler**.
   g. Select **Custom** from the Keystore menu.
   h. Click **Custom keystore configuration**.
   i. Enter the keystore path. For example: `${USER_INSTALL_ROOT}/etc/ws-security/samples/enc-receiver.jceks`.
   j. Select **JCEKS** for the Type.
   k. Enter the password in the Password and Confirm password fields. For example, **storepass**.
   l. Enter a Key Name. For example, **CN=Bob, O=IBM, C=US**.
   m. Enter a Key Alias. For example, **bob**.
   n. Click **OK**.
   o. Click **OK** again.
   p. Click **OK** one more time to return to the **Enterprise Applications > WSSampleServicesSei > Service client policy sets and bindings > *binding_name* > WS-Security > Authentication and protection** panel.
   q. The status of AsymmetricBindingRecipientEncryptionToken0 should display as **Configured**.
2. Modify the encrypted parts settings for the client binding, as follows:
   a. Click **request:app_encparts** under Request message signature and encryption protection.
   b. Enter a Name. For example, **MyEncPart**.
   c. Click **New** under Key information.
   d. Fill in a Name. For example, **MyEncKeyInfo**.
   e. Click **OK**.
   f. Select **MyEncKeyInfo** (or the name that you specified for the encrypted part) from the Available box and click **Add**. MyEncKeyInfo appears in the **Assigned** box.
   g. Click **OK** to return to the **Enterprise Applications > WSSampleServicesSei > Service client policy sets and bindings > *binding_name* > WS-Security > Authentication and protection** panel.
   h. The status of request:app_encparts should display as **Configured**.
3. Configure the Username token settings in the client binding, as follows:
   a. Click **request:myUserNameToken** under Authentication tokens.
   b. Click **Apply**.
   c. Click **Callback handler**.
   d. Specify the User name. For example, **LDAPSunuser6**.
   e. Specify the password, and confirm the password.
   f. Click **OK**.
   g. Under **Custom properties**, click **New** to add the properties for enabling nonce and timestamp.

h. Enter the property name `com.ibm.wsspi.wssecurity.token.username.addNonce` to enable nonce, and the property value `true`.

i. Enter the property name `com.ibm.wsspi.wssecurity.token.username.addTimestamp` to enable timestamp, and the property value `true`.

j. Click **OK** again.

k. The status of request:myUserNameToken should now display as **Configured**.

l. Click **Save** to save your client bindings.

4. Configure the server binding, as follows:

a. Attach the policy to a service resource and create a new binding for that resource that includes the WSSecurity policy.

b. Click on **WSSecurity** in the new binding to display the main WSSecurity binding panel. For example, click **Enterprise Applications > WSSampleServiceSei > Service client policy sets and bindings > *binding_name* > WS-Security**.

c. Click **Authentication and protection**.

d. Under Protection tokens, click AsymmetricBindingRecipientEncryptionToken0 (the Asymmetric encryption consumer)

e. Click **Apply**.

f. Click **Callback handler**.

g. Select **Custom** from the Keystore menu.

h. Click **Custom keystore configuration**.

i. Enter the keystore path. For example: `${USER_INSTALL_ROOT}/etc/ws-security/samples/enc-receiver.jceks`.

j. Select **JCEKS** for the Type.

k. Enter the password in the Password and Confirm password fields. For example, **storepass**.

l. Enter a Key Name. For example, **CN=Bob, O=IBM, C=US**.

m. Enter a Key Alias. For example, **bob**.

n. Enter the password for the keypass in the Password and Confirm password fields.

o. Click **OK**.

p. Click **OK** again.

q. Click **OK** one more time to get return to the **Enterprise Applications > WSSampleServicesSei > Service client policy sets and bindings > *binding_name* > WS-Security > Authentication and protection** panel.

r. The status of AsymmetricBindingRecipientEncryptionToken0 should display as **Configured**.

5. Modify the encrypted parts settings for the server binding, as follows:

a. Click **request:app_encparts** under Request message signature and encryption protection.

b. Enter a Name. For example, **MyEncPart**.

c. Click **New** under Key information.

d. Fill in a Name. For example, **MyEncKeyInfo**.

e. Click **OK**.

f. Select **MyEncKeyInfo** (or the name that you specified for the encrypted part) from the Available box and click **Add**. MyEncKeyInfo appears in the **Assigned** box.

g. Click **OK** to return to the **Enterprise Applications > WSSampleServicesSei > Service client policy sets and bindings > *binding_name* > WS-Security > Authentication and protection** panel.

h. The status of request:app_encparts should display as **Configured**.

6. Configure the Username token settings in the server binding, as follows:

a. Click **request:myUserNameToken** under Authentication tokens.

b. Click **Apply**.
c. Click **Callback handler**.
d. Click **OK**.
e. Under **Custom properties**, click **New** to add the properties for verifying nonce and timestamp.
f. Enter the property name `com.ibm.wsspi.wssecurity.token.username.verifyNonce` to verify nonce, and the property value `true`.
g. Enter the property name `com.ibm.wsspi.wssecurity.token.username.verifyTimestamp` to verify timestamp, and the property value `true`.
h. Click **OK** again.
i. The status of request:myUserNameToken should display as **Configured**.
j. Click **Save** to save the server bindings.

Configuring the username and password for WS-Security Username or LTPA token authentication

When using the Username WSSecurity default policy set, you must configure the username and password for username token authentication separately from the security settings defined in the bindings.

About this task

When you install a JAX-WS application and attach the default Username WSSecurity default policy set, the next step is to configure the general provider sample binding for the JAX-WS provider, and the general client sample binding for the JAX-WS client. However, the binding file for the default client sample binding does not include a username or password for token authentication. Since the username and password is not available from the target deployed system, you must specify a valid username and password in your environment using the administrative console.

Procedure

1. Log in to the administrative console, then click **Services** > **Policy sets** > **General client policy set bindings**.
2. Click **Client sample** to edit the binding.
3. Click **WS-Security**.

   Add basic authentication information, such as username and password, to the general client sample bindings for any policy set that uses a Username token or LTPA token, including:
   - Username SecureConversation
   - Username WS-I RSP
   - LTPA SecureConversation
   - LTPA WS-I RSP
   - LTPA WSSecurity default
4. Click **Authentication and protection**.
5. In the Authentication tokens table, click **gen_signunametoken** to edit the username token settings.
6. Click **Callback handler** in the Additional Bindings section.
7. Enter the appropriate username and password information for your environment in the **User name** and **Password** fields.
8. Enter the password a second time in the **Confirm Password** field, then click **Apply**.
9. Repeat steps 5 through 8 for the gen_signltpatoken LTPA token generator.

Results

Note: This administrative console panel applies only to Java™ API for XML Web Services (JAX-WS) web services.

Securing requests to the trust service using system policy sets

WebSphere Application Server provides message-level protection for its security token service, known as the WebSphere Application Server trust service. For the trust service, you must use a special class of policy sets known as system policy sets.

Before you begin

You can secure requests to the trust service by using two different configuration methods:

- Use the administrative console to define and attach a system policy set and binding to a trust service operation that is associated with an endpoint.
- Use the wsadmin tool, which supports the Jython and Jacl scripting languages, to configure system policy sets for the trust service. You can manage the policies for the Quality of Service (QoS) by creating policy sets and managing associated policies.

About this task

For WebSphere Application Server trust service security, you must configure the system policy sets, the bindings, the trust service attachments, and the security cache.

Perform the following high-level steps. The order of the tasks is not important but all high-level required steps must be performed to complete the trust configuration.

Procedure

1. Define a new system policy set or manage existing system policy sets. To manage system policy sets, you can perform the following tasks:
   a. Define the system policy set and binding. The system policy set can be a new or existing policy set. If you create a new system policy set, you must specify and configure the policy types. A default binding configuration is associated with each policy type.
   b. Modify the system policy set, as needed.

      Other optional policy set-related tasks that you can perform include:

      - Add, edit, or remove policy set attachments.
      - Edit, enable, disable or remove policy types
      - Create a system policy set by selecting and copying an existing system policy set. When copying an existing system policy set, you also specify whether to move the existing attachments to this new system policy set.
      - Delete system policy sets. You cannot delete pre-configured system policy sets that are provided by WebSphere Application Server by default.
      - Archive a system policy set by selecting and exporting an existing system policy set. When exporting an existing system policy set, you create a .zip archive file. The .zip file for exporting the policy set is provided for downloading. For example, if you have a policy set named ABC_ps and you want to export and move the archive file from ServerA to ServerB, first use the export function to create the .zip file. Then, manually transfer the archive file to ServerB.
2. Create and manage explicit attachments. You can perform the following trust service attachment tasks:
   a. Attach the system policy set and assign a binding to an endpoint. For an endpoint, you can create explicit attachments for each of the four trust service operations to the respective Trust Service Defaults policy sets and bindings. After you have created these initial attachments, you can view and further modify existing policy set and binding configurations.
   b. Modify existing policy set attachment and binding configurations, as needed.. The system policy set can be a new or existing policy set. If you create a new system policy set, you must specify and configure the policy types. A default binding configuration is associated with each policy type.

      The system policy set that is attached to issue and renew must correspond to the client and endpoint's bootstrap policy set and the system policy set attached to validate and cancel must

correspond to the client and endpoint's application policy set. The bootstrap policy set for the endpoint service is only required if the endpoint service makes issue and renew requests to the trust service.

Other optional attachment-related tasks that you can perform include:

- Change the system policy set and binding configurations.
- Create custom system policy sets and bindings.
- Attach each of the four default trust service operations to a system policy set and binding.
- Attach each of the four trust service operations associated with a specific endpoint to a system policy set and binding.
- Specify that the selected trust service operations for an endpoint inherit the respective default trust service policy set and binding.
- Assign the Default binding or a custom binding configuration to the selected policy set attachment.
- Update the trust service runtime configuration.

3. Manage the security context token provider that the trust service provides. You can perform the following trust service token provider tasks:
   a. Modify the configuration of the Security Context Token provider, as needed..

      Other optional token provider-related tasks that you can perform include:

      - Update the trust service runtime configuration for any token provider configuration changes.
4. Manage the trust service default token provider and any endpoints that have an explicitly assigned token (rather than inheriting from the default). Targets are endpoints that are assigned a specific token provider. You can perform the following trust service target tasks:
   a. Create a new trust service target by explicitly assigning a service endpoint URL to the default token provider.. Performing this task creates an explicit assignment to the default trust service token provider, the Security Context Token. All other endpoints inherit the trust service default token provider.
   b. Configure a target. WebSphere Application Server defines one default supported token provider, the Security Context Token. Other tasks that you can perform for existing targets include:
      - Modifying one or more endpoints that have a security context token provider explicitly assigned.
      - Changing the token provider for an endpoint from inherited to explicitly assigned. Therefore, the token provider for the endpoint does not change as the default trust service token provider changes.
      - Changing the token provider for an endpoint from explicitly assigned to inherited. Therefore, the token provider for the endpoint is the default trust service token provider and changes as the default changes.
      - Updating the trust service runtime configuration.
5. Configure the security cache. You can change the behavior of client-side security caching.
6. Update the trust service runtime configuration. You must update the runtime configuration whenever one or all of the following trust-related items are created or changed:
   - Trust service attachments
   - Token providers
   - Targets

Results

After the configurations are completed and the trust service runtime configuration has been updated, you have used the administrative console to secure requests to the trust service by using system policy sets.

Enabling secure conversation:

Use secure conversation to secure web services application messages.

Before you begin

Applications that contain web services must have been deployed.

About this task

The Organization for the Advancement of Structured Information Standards (OASIS) Web Services Secure Conversation (WS-SecureConversation) draft specification describes ways to establish a secure session between the initiator and recipient of SOAP messages. The WS-SecureConversation draft specification also defines how to use the OASIS Web Services Trust (WS-Trust) protocol to establish a security context token (SCT). For complete information, see the OASIS Web Services Secure Conversation specification.

WebSphere Application Server supports the ability of an endpoint to issue a security context token for WS-SecureConversation, and thereby provides a secure session between the initiator and recipient of SOAP messages.

The following figure describes the flow that is required to establish a secured context and to use session-based security.

Figure 45. Displaying the flow between the client and the web service and security token service

In the WS-SecureConversation specification, a security context is represented by the <wsc:SecurityContextToken> security token. The following example represents the assertion syntax for a <wsc:SecurityContextToken> element.

```
<wsc:SecurityContextToken wsu:Id="..." ...>
    <wsc:Identifier>...</wsc:Identifier>
    <wsc:Instance>...</wsc:Instance>
    ...
</wsc:SecurityContextToken>
```

The security context token does not support references to it by using key identifiers or key names. All references must either use an ID (to a wsu:Id attribute) or a <wsse:Reference> to the <wsc:Identifier> element.

WebSphere Application Server provides these pre-configured secure conversation-related polices:

- The **SecureConveration** policy set follows the WS-SecureConversation and WS-Security specifications and provides a policy set with secure conversation enabled and using keys derived from security context token for signing and encrypting the application messages.
- The **Username SecureConversation** policy set follows the WS-SecureConversation and WS-Security specifications and adds authentication using the Username token.
- The **LTPA SecureConversation** policy follows the WS-SecureConversation and WS-Security specifications and provides authentication using the Lightweight Third Party Authentication (LTPA) tokens.

In this example, the default SecureConversation policy set, and the default WS-Security binding and TrustServiceSecurityDefault binding are used to achieve the task of enabling secure conversation. The default SecureConversation policy set has both the application policy (symmetricBinding) and the bootstrap policy (asymmetricBinding). The application policy is used to secure application messages and the bootstrap policy is used to secure the RequestSecurityToken (RST) messages.

A trust service that issues a security context token is configured with the TrustServiceSecurityDefault system policy and the TrustServiceSecurityDefault binding. The trust policy is responsible for securing RequestSecurityTokenResponse (RSTR) messages. If the bootstrap policy is modified, the trust policy has to be modified to match both of the configurations.

Note: The following steps are to be used only in development and test environments.

The Web Services Security (WS-Security) default bindings that are used here contain sample key files and must be customized before use in a production. For the production environment, use of custom bindings is advised. Also note that, if the profile is created by using the choice of **Create the server using the development template**, you can skip steps 2 and 3.

To configure secure conversation, configure the policy set, and add a policy assertion to the policy, complete the following steps:

Procedure

1. Make a copy of a default secure conversation policy so you can customize the policy set for your own environment.
   a. Launch the administrative console, and click **Services** > **Policy sets** > **Application policy sets**.
   b. Select the check box next to an existing policy set that follows the WS-SecureConversation specifications. For example, you might click the check box next to **SecureConversation**. This policy set is one of the pre-configured secure conversation-related application policy sets that is listed in the table. The SecureConversation policy set has a bootstrap policy to match the default policy set for the trust service to issue and renew tokens.
   c. Click **Copy**.
   d. Enter a unique name for the new copy of the SecureConversation application policy set. For example: CopyOfSCPolicySet
   e. Optional: Change the description, as needed, for your customized version of this policy set.
2. Attach the policy set and binding to the application.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application name***.
   b. Click either **Service provider policy sets and bindings** or **Service client policy sets and bindings** to attach resources to the CopyOfSCPolicySet policy set. The general binding is assigned automatically as the default.
   c. You can use the **Attach Policy Set** and **Assign Binding** menu lists to select a different policy set or binding.

Results

After completing these steps, you have configured secure conversation.

What to do next

Next, review the example scenario about how to establish a security context token to secure a secure conversation.

Web Services Secure Conversation:

Web Services Secure Conversation (WS-SecureConversation) provides a secured session for long running message exchanges and leveraging of the symmetric cryptographic algorithm.

WS-SecureConversation provides session-based security. Session-based security optimizes long message exchanges, as symmetric cryptography can be used to sign and encrypt the message. Typically, symmetric

cryptographic algorithm is less CPU intensive than the asymmetric cryptography. Symmetric cryptographic algorithms should provide better performance and throughput when compared to the asymmetric cryptographic algorithms.

The symmetric cryptographic algorithm also provides a means to secure other session-based protocol and exchange patterns, such as Web Services Reliable Messaging (WS-ReliableMessaging).

Security context token for secure conversation

The Web Services Security specification defines the basic mechanisms for providing secure messaging. The Web Services Trust (WS-Trust) specification defines extensions to Web Services Security that provide ways to establish and broker trust relationships between two parties. The WS-Trust protocol defines the syntax of the request that can be sent to a security token service and the corresponding or subsequent response of the security token service. The security token service provided with WebSphere Application Server is called the *trust service*.

Using the WS-Trust protocol, a party can request the trust service issue a security context token (SCT). Then, this token can be used to establish a secure conversation (WS-SecureConversation). The request for a security token is sent to an application endpoint. The request is intercepted by the WebSphere Application Server and routed to the trust service.

A policy can be defined as the default for all trust issue operations, renew operations, validate operations, or cancel operations. Additionally, a policy can be attached to a specific URL and operation pair.

WS-SecureConversation defines extensions to allow security context establishment and sharing, and session key derivation, which allows contexts to be established and, potentially, more efficient keys, or new key material, to be exchanged. The WebSphere Application Server support for WS-Trust and WS-SecureConversation focuses on the issuing, renewing, validating, and cancelling of the security context token for secure conversation.

Policy set and bootstrap policy

In addition to describing these functions, the OASIS WS-SecureConversation draft submission describes multiple methods of establishing a secure session between the initiator and the recipient of the SOAP messages.

The bootstrap security policy is the security policy for the initiating party to acquire the security token for secure conversation from the trust service by using a token-issuing WS-Trust or WS-SecureConversation protocol message. The policy set configuration consists of the security policy for communication with the application service, and the bootstrap policy for communication with the trust service.

If sharing of a policy configuration (using WS-Policy) containing the secure conversation bootstrap policy fails, it may be because the bootstrap request and response policies differ. The message part protection for the bootstrap policy must be the same for both request and response bootstrap messages, because a single policy is published for both request and response.

What is supported for Web Services Secure Conversation

The following list highlights some of the key functions that are supported in WebSphere Application Server. The list is not exhaustive.

- A security context token (SCT) established between the initiating party and the recipient party.
- The WS-SecureConversation operations that are supported on the security context token (SCT), such as Issue token, Renew token, and Cancel token. Validate token is supported using WS-Trust protocol.
- A derived key (explicit and implied)

What is not supported for Web Services Secure Conversation

The following list highlights some of the key functions that are not supported in WebSphere Application Server. The list is not exhaustive.

- WS-SecureConversation does not support establishing a security context through the security context token that is created by an external security token service (trust component). However, WebSphere Application Server supports an internal security token service.
- WebSphere Application Server does not support establishing a security context through the security context token that is created by one of the communicating parties and propagated with a message.
- WebSphere Application Server does not support amending a security context token.
- WebSphere Application Server does not support a client creating the security context token.
- WebSphere Application Server provides no support for exchange and negotiation.

Secure conversation scenarios

The following scenarios describe the WS-SecureConversation functions that WebSphere Application Server supports:

- WS-SecureConversation

  This scenario is based on establishing a security context token with the recipient and using the derived key to sign and encrypt the message. It describes how to establish a security context by using session-based security. Session-based security is where the flow of the initiator establishes the security context token by using the WS-SecureConversation protocol with the recipient.
- WS-SecureConversation with WS-ReliableMessaging

  This scenario is a composite scenario that includes functions that are required for the composition scenario of Web Services Reliable Messaging (WS-ReliableMessaging), WS-SecureConversation, and WS-Trust. This scenario describes how to use WS-SecureConversation with WS-ReliableMessaging where the flow is similar to the previous scenario, but which is from the secure conversation prospective. However, the main difference is that the WS-ReliableMessaging sequence is secured with the security context token and scopes the WS-ReliableMessaging sequence to the security context token. This description focuses on the message exchanges that are using the security context token in the overall flow.

Scoping of Web Services Secure Conversation:

Web Services Secure Conversation supports two scoping mechanisms: the default and the Java API for XML Web Services (JAX-WS) client service level.

Review the following information about the two scoping mechanisms to ensure the proper scoping of secure conversation and policy set for WebSphere Application Server.

Default

The default scope is based on a cluster, an application, a module, and a target service endpoint. For a client running in a thin client environment, it is considered to be a single application, cluster, and module.

In this scoping mode, all the instances of the JAX-WS client within a particular application, cluster, and module to the same target service endpoint share the same secure conversation. For example, in the following figure, the two client instances (Client 1 and Client 2) are in the same module. Client 1 and Client 2 share the same secured conversation with Service 1. The other two client instances (Client 3 and Client 4), which are in a different module than Clients 1 and 2 and which share a secured conversation with each other but not with Clients 1 and 2.

JAX-WS client service level

Scope at the JAX-WS client service level is enabled by specifying a property in the token generator binding configuration of the Secure Conversation Token (SCT)in the client application request (application outbound). The binding is located in the META-INF of the deployed application.

For example, if the application is `WSSampleClientBeta.ear` and the binding directory is `SecureConversation123binding`, the binding file would be located at:

```
$PROFILE_DIR/config/cells/<cellname>/WSSampleClientBeta.ear/deployments/WSSampleClientBeta
/META-INF/SecureConversation123binding/PolicyTypes/WSSecurity/bindings.xml.
```

An example of the configuration follows:

```
<tokenGenerator name="gen_enctgen"
      classname="com.ibm.ws.wssecurity.wssapi.token.impl.CommonTokenGenerator">
   <valueType localName="http://schemas.xmlsoap.org/ws/2005/02/sc/sct" uri="" />
      <callbackHandler classname="com.ibm.ws.wssecurity.impl.auth.callback.WSTrustCallbackHandler">
         <properties name="com.ibm.ws.wssecurity.sc.SCTScope" value="SERVICE_SCOPE"/>
      </callbackHandler>
         <properties name="com.ibm.ws.wssecurity.sc.dkt.ServiceLabel" value="WSC"/>
         <properties name="com.ibm.ws.wssecurity.sc.dkt.ClientLabel" value="WSC"/>
            <jAASConfig configName="system.wss.generate.sct"/>
</tokenGenerator>
```

The following code example demonstrate the behavior after the property in the token generator binding configuration of the SCT in the client application request (application outbound) is enabled. In this mode, Web Services Secure Conversation is scoped at the JAX-WS client service instance.

```
QName serviceQname = new QName("http://ws.apache.org/axis2", "EchoService");
QName portQname = new QName("http://ws.apache.org/axis2", "EchoServicePort");
String endpointUrl = "http://myhost/......";
Service svc1 = Service.create(serviceQname);
svc1.addPort(portQname, null, endpointUrl);
Dispatch<Source> dispatch = svc1.createDispatch(portQname, Source.class, null);
......
......
Service svc2 = Service.create(serviceQname);
svc2.addPort(portQname, null, endpointUrl);
Dispatch<Source> dispatch = svc2.createDispatch(portQname, Source.class, null);
```

where `svc1` and `svc2` are in two different secure conversations with the target service endpoint.

You can change the scope by using either the administrative console or by using scripting to add a property.

Secure conversation client cache and trust service configuration:

For both distributed and local clients, the WebSphere Application Server secure conversation client cache stores tokens on the client.

WebSphere Application Server supports caching of the security context token for both the distributed client and local client. If the security context token is distributed, a client in the same replication domain uses the same security context token. Distributed caching also supports disk offload to save the security context token to disk for recovery. When the client runs applications using secure conversation, and is part of a cluster setup, then the client can use the distributed cache mechanism to replicate the token data among the cluster members.

To use the administrative console to modify the cache settings, click **Services** > **Security Cache**.

You can configure the cache settings, such as the following.

- Set the time that the token remains in the cache after timeout. The default value is 10 minutes. This value is a time window to renew an expired token.

- Set the renewal interval before the token expires. The default value is 10 minutes, and the minimum value is 3 minutes. Entering a number less than 3 minutes causes an error.

  Important: This setting is critical. This setting represents the maximum roundtrip time for a client to make a request, the transport request to go to the server, the server to process the request, and the transport response (if applicable) back to the client. If the time specified is too small and there is not enough time specified, then the token might expire during the roundtrip, and the client receives a failure response. If the time specified is too large, then performance diminishes.

  If the security context token is renewed too often, it might cause Web Services Secure Conversation (WS-SecureConversation) to fail or even cause an out-of-memory error to occur. It is required that you set the renewal interval before the token expires value for the Secure conversation client cache to a value less than the token timeout value for the security context token. It is also suggested that the token timeout value be at least two times the renewal interval before the token expires value.
- Select the **Enable distributed caching** check box to support distributed clients. You must ensure that the WebSphere Application Server dynamic cache service, and cache replication, are enabled. For more information on enabling the dynamic cache service, refer to the topic Enabling the distributed cache using synchronous update and token recovery.
- Define a custom property, edit, or remove existing custom properties.

The WS-SecureConversation client rejects a security context token that is issued at a future time. If you cannot synchronize the clock between the client machine and service machine, the clock skew could be configured to prevent the rejection of a valid token. The default clock skew is 3 minutes. To modify the default clock skew setting, add the following custom property to the desired minutes:

```
clockSkewToleranceInMinutes
```

Alternatively, use the wsadmin commands to manage secure conversation client cache configurations.

Thin client

For a web service application client running outside WebSphere Application Server, the security context token is cached only in the local Java process. The following system properties can be used to override the default cache setting on the thin client:

com.ibm.wsspi.wssecurity.SC.cache.cushion
: Specifies the time in minutes to renew a security context token to be used with WS-SecureConversation on the client side so that the security context token has enough time to complete the downstream call. The default value is 10 minutes, and the minimum value is 3 minutes.

com.ibm.wsspi.wssecurity.SC.token.clockSkewTolerance
: Specifies the tolerant clock skew time for a token between two machines. The default value is 3 minutes.

WS-Reliable Messaging settings

When WebSphere Application Server applications use policies such as WS-I RSP with managed persistent WS-Reliable Messaging, modify the cache and trust configuration values.

Set the cache configuration time value to 120 minutes.

1. In the WebSphere Application Server administrative console, click **Services** > **Security Cache**.
2. Modify the value of the **Time token is in cache after timeout** field from 10 to 120.
3. Click **Apply**, and then click **Save**.

Increasing the cache time value means that the token remains in the cache for a longer period after token expiration, so that the token is available for renewal. The WS-Reliable Messaging runtime scopes the

CreateSequence message to the security context token. Therefore, it is important to maintain the same security context for the life time of the Reliable Messaging sequence.

Enable distributed caching using the default option, Synchronous update of cluster members, to support distributed clients. For more information, refer to the topic Enabling the distributed cache using synchronous update and token recovery.

Additional recommended changes

Other important configuration changes are also recommended.

- Modify the life time of the Security Context Token by changing the value from the default of 120 minutes, to 600 minutes.
- Modify the Renew after expiration value by changing the value from `false` to `true`.
- Modify settings for the token providers, as follows:
  1. In the administrative console, click on **Services** > **Trust service** > **Token providers**.
  2. Click **Security Context Token**.
  3. Change the value in the **Token timeout** field from 120 to 600.
  4. Click the check box to select **Allow renewal after timeout**.
  5. Click **Apply**, and then click **Save**.

Derived key token:

After establishing the security context and after the secret have been established (authenticated), derived keys can be used to sign and encrypt the SOAP message to provide message level protection. You can then use derived keys for each key that is used in the security context.

You can enable Web Services Secure Conversation (WS-SecureConversation) by using symmetric keys that are derived from the security token for signing and encrypting the application messages.

Using WS-SecureConversation, the initiator can establish a security context token using the Web Services Trust (WS-Trust) protocol with the recipient. A security context token implies or contains a shared secret. Using a common secret, different key derivations can be defined. Then, using the security context token, the <wsc:DerivedKeyToken> token can be used to derive keys from any security token that has a shared secret, key, or key material. This secret can be used for signing or encrypting messages, but it is recommended that derived keys be used for signing and encrypting messages that are associated only with the security context.

Syntax for the <wsc:DerivedKeyToken> element

The <wsc:DerivedKeyToken> element is used to indicate that the key for a specific reference is generated from the function so that explicit security tokens, secrets, or key material need not be exchanged as often. The derived key token does not support references to it using key identifiers or key names. All references must use an ID to a *wsu:Id* attribute or use a URI reference, <wsse:Reference>, to the <wsc:Identifier> element in the security context token.

The syntax for <wsc:DerivedKeyToken> element is as follows:

```
<wsc:DerivedKeyToken wsu:Id="...">
    <wsse:SecurityTokenReference>...</wsse:SecurityTokenReference>
    <wsc:Label>...</wsc:Label>
    <wsc:Nonce>...</wsc:Nonce>
  </wsc:DerivedKeyToken>
```

Derived keys are expressed as security tokens and use different algorithms for deriving keys. The following URI is used to represent the derived key token type:

```
http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/dk
```

The nonce is processed as a binary octet sequence (the value prior to base64 encoding). The nonce seed is required, and must be generated by one or more of the communicating parties. Use separate nonces and have independently generated keys for signing and encrypting for request and response. New keys should be derived for each message, meaning that a previous nonce should not be reused.

Implied derived key generation

Implied derived keys define a shortcut mechanism for referencing certain types of derived keys. Specifically, an @wsc:Nonce attribute can be added to the security token reference (STR) that is defined in the WS-Security specification. When present, an implied derived key indicates that the key is not in the referenced token but, instead, is a key that is derived from the key or secret of the referenced token. It is recommended that you do not use implied derived Keys in the <wsc:DerivedKeyToken> element.

The following example illustrates a message that is sent using two derived keys, one for signing and one for encrypting:

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
    xmlns:xenc="..." xmlns:wsc="..." xmlns:ds="...">
  <S11:Header>
    <wsse:Security>
      <wsc:SecurityContextToken wsu:Id="ctx2">
        <wsc:Identifier>uuid:...UUID2...</wsc:Identifier>
      </wsc:SecurityContextToken>
      <wsc:DerivedKeyToken wsu:Id="dk2">
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#ctx2"/>
        </wsse:SecurityTokenReference>
        <wsc:Nonce>KJHFRE...</wsc:Nonce>
      </wsc:DerivedKeyToken>
      <xenc:ReferenceList>
        ...
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#dk2"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
          ...
      </xenc:ReferenceList>
      <wsc:SecurityContextToken wsu:Id="ctx1">
        <wsc:Identifier>uuid:...UUID1...</wsc:Identifier>
      </wsc:SecurityContextToken>
      <wsc:DerivedKeyToken wsu:Id="dk1">
        <wsse:SecurityTokenReference>
          <wsse:Reference URI="#ctx1"/>
        </wsse:SecurityTokenReference>
        <wsc:Nonce>KJHFRE...</wsc:Nonce>
      </wsc:DerivedKeyToken>
      <xenc:ReferenceList>
        ...
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#dk1"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
        ...
      </xenc:ReferenceList>
    </wsse:Security>
  ...
  </S11:Header>
  <S11:Body>
    ...
  </S11:Body>
</S11:Envelope>
```

Enabling secure conversation in a mixed cluster environment:

When Web Services Security applications using secure conversation run in a mixed cluster environment, an interoperability property must be set for the WebSphere Application Server Version 8.0 nodes.

About this task

A mixed cluster environment consists of nodes running WebSphere Application Server Version 6.1 Feature Pack for Web Services, and nodes running WebSphere Application Server V7 and later. To run Web Services Security applications utilizing secure conversation in this environment, enable the following

property for the V7.0 and later nodes in the cluster:
com.ibm.ws.wssecurity.distributedcache.PreV70InteropMode.

This property ensures that the method for adding entries in the cache is consistent between the different nodes, and also allows the applications to interoperate. To enable the property, follow these steps:

Procedure

1. In the WebSphere Application Server administrative console, click **Servers** > **Server Types** > **WebSphere application servers**.
2. Click the server name.
3. Under the Security section, click **JAX-WS and JAX-RPC security runtime**.
4. Click **Custom properties**.
5. Click **New**.
6. Enter the property name, `com.ibm.ws.wssecurity.distributedcache.PreV70InteropMode`, in the Property name field. Enter the property value of `true` in the Property value field, then click **OK**.
7. When the Properties panel is refreshed, the com.ibm.ws.wssecurity.distributedcache.PreV70InteropMode property appears in the properties table.
8. Click **Save** to commit the change.
9. Restart the server to load the new property.

Results

Secure conversation is now enabled for Web Services Security applications running in a mixed cluster environment.

Enabling distributed cache and session affinity when using Secure Conversation:

WebSphere Application Server provides message-level protection in a cluster environment. You can use Web Services Secure Conversation (WS-SecureConversation) for message-level protection of Java API for XML Web Services 2.0 (JAX-WS) Web services in a cluster environment.

Before you begin

A web services request that is protected with a Security Context Token (SCT) is routed to one server in a cluster, but that SCT might have been issued or renewed by a different server in the cluster. If the WebSphere Application Server distributed cache is not configured to replicate or does not replicate quickly enough, the server processing the request might not have access to the SCT. The task steps described in this topic need to be performed only if the replication setting for cluster members is set to asynchronous update for the Web Services Security distributed cache.

For more information on cache update settings, read the topic Enabling the distributed cache using synchronous update and token recovery. You can also enable the Web Services Security distributed cache with the default setting, which enables synchronous update of cluster members.

About this task

Perform the following high-level steps to enable distributed cache and session affinity when using secure conversation for message-level protection in a cluster environment.

Procedure

1. Enable the distributed cache for the Security Context Token.
   a. In the administrative console for WebSphere Application Server, click **Services** > **Security cache**.
   b. Select the **Enable distributed caching** check box.

c. Click the radio button to select **Asynchronous update of cluster members**.
d. Click **Apply** and then click **Save** to save the configuration.

2. Create a replication domain. Perform the following steps:
   a. In the Administrative Console, click **Environment** > **Replication domains** > **New**.
   b. Enter a name. For example, `ABCDomain`.
   c. Under Number of replicas, select the **Entire Domain** option.
   d. Click **OK** and then click **Save** to save the configuration.
3. Enable the dynamic cache. Perform the following steps for each server in the cluster:
   a. In the Administrative Console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Container Services** > **Dynamic Cache Service**.
   b. Select the **Enable cache replication** option.
   c. Select the replication domain name that you created. For example, `ABCDomain`.
   d. Select the replication type as **Both push and pull**.
   e. Click **OK** and then click **Save** to save the configuration.
4. Optional: Change the distributed cache batch update interval. By default, the distributed cache batch update interval is 1,000 milliseconds. However, you can set this interval to a value that is less than 1,000 milliseconds. To change the value, complete the following steps for each server in the cluster:
   a. In the Administrative Console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Java and Process Management** > **Process Definition** > **Java Virtual Machine** > **Custom Properties** > **New**.
   b. Enter the `com.ibm.ws.cache.CacheConfig.batchUpdateInterval` property name.
   c. Enter the property value.
   d. Click **OK** and then click **Save** to save the configuration.
5. Install and configure a web server or proxy server that supports session affinity. The IBM HTTP Server and WebSphere Application Server proxy server support session affinity. In the WebSphere Application Server Information Center, read the topic "Communicating with Web servers." for information on installing and configuring the IBM HTTP Server.
6. Configure the client systems to send the web services requests to the host and port where the web server or proxy server is running. The web server or proxy server then routes the requests to the proper cluster member.
7. On the services that are receiving the web services requests, which are protected by using Web Services Secure Conversation, select the HTTP transport Session enabled policy option. Complete the policy set configuration by following these steps:
   a. Add the HTTP Transport policy to the policy set that is being used by the services.
   b. In the configuration panel for the HTTP Transport policy, select **Session enabled**.
   c. Click **OK** and then click **Save** to save the configuration.
8. On the client systems that are sending the web services requests and are protected by Secure Conversation, enable the HTTP transport Maintain session property. Complete the policy set configuration or set the property programmatically. If you are using a policy set with your configuration, follow these steps:
   a. Add the HTTP Transport policy to the policy set that is being used by the clients.
   b. At the HTTP Transport policy configuration panel, select the **Session enabled** option.
   c. Click **OK** and then click **Save** to save the configuration.

Results

After the configurations are completed, you have enabled the distributed cache and session affinity when using secure conversation in a cluster environment. If the server processing the request does not have access to the SCT, it will fail the request with the error of `Either null SCT or invalid SCT`.

Example

The following example, which is a code snippet, demonstrates how to programmatically set the Maintain session property on the correct JAX-WS object:

```
Map<String> rc = ((BindingProvider) port).getRequestContext();
...
rc.put(BindingProvider.SESSION_MAINTAIN_PROPERTY, Boolean.TRUE);
... </String>
```

Flow for establishing a security context token to secure conversations:

This example scenario describes the flow of how the initiator establishes the security context token (SCT) by using the WS-Trust protocol for session-based security with the recipient. After establishing the security context token, derived keys from the security context token are used to sign and encrypt the SOAP message to provide message-level protection. This examples focuses on the message exchanges using the security context token in the overall flow of the SOAP messages.

The Organization for the Advancement of Structured Information Standards (OASIS) Web Services Secure Conversation (WS-SecureConversation) specification describes ways to establish a secure session between the initiator and recipient of SOAP messages. The WS-SecureConversation specification also defines how to use Web Services Trust (WS-Trust) protocol to establish a security context token. The product supports both Version 1.3, and the draft version, of the WS-SecureConversation specification.

WebSphere Application Server supports the ability of an endpoint to issue a security context token for WS-SecureConversation and thereby provides a secure session between the initiator and recipient of SOAP messages.

The following figure describes the flow that is required to establish a secured context and to use session-based security.

Figure 46. Displaying the flow between the client and the Web service and security token service

Exchanging messages between the initiator and the recipient

The following figure shows how the messages are exchanged between the initiator and the recipient to establish the security context token. The two WS-Trust protocols, RequestSecurityToken (RST) and RequestSecurityTokenResponse (RSTR), are used to request the security context token from the recipient endpoint.

The bootstrap policy is used to secure the RST and validate the RSTR request, which is typically different from the application security policy.

Figure 47. Using WS-Trust protocols RST and RSTR to establish the SCT between the initiator and the recipient

Scenario describing how to use secure conversation

Typically, to use secure conversation, the following steps are involved;

1. The client sends a RequestSecurityToken (RST) trust request for a security context token to an application endpoint with its secret key (entropy and target key size) and requests the target service secret key.

   This request is typically secured with asymmetric Web Service Security that is defined in the bootstrap policy.

2. The RST is processed by the trust service and, if the request is trusted based on the security policy, the trust service returns the security context token with the target service secret key by using a WS-Trust RequestSecurityTokenResponse (RSTR).

   This request is typically secured with asymmetric Web Service Security. The client verifies whether the RSTR can be trusted, based on the bootstrap policy.
3. If the RequestSecurityTokenResponse is trusted, the client secures (signs and encrypts) the subsequent application messages by using the session keys.

   The session keys are derived from secret of the security context token that is obtained from the initial WS-Trust RequestSecurityToken and RequestSecurityTokenResponse messages that are exchanged between the initiator and the recipient.
4. The specification defines an algorithm of how to derive the key based on the initial secret. The target web service calculates the derive key from the metadata contained in the security header of the SOAP message and the initial secret.
5. The target web service uses the derived key to verify and decrypt the message based on the application security policy.
6. The target web service uses the derived key from the secret to sign and encrypt the response based on the application security policy.
7. Repeat of steps 3 through 6 until the message exchange has completed.

Using keys that are derived from the secret of the security context token

After the security context token is established, the application messages are secured with message protection by using keys that are derived from the secret of the security context token. The derived keys are used to secure the application messages by signing and encrypting the application messages. The security context token contains a UUID, which is used as identification of a shared secret. The token UUID can be used in the SOAP message to identify the security context token for the message exchanges. The secret must be kept in memory by the session participants (in this case the initiator and the recipient) and protected. Compromising the secret undermines the secure conversation between the participants.

Figure 48. Securing application messages with keys derived from secret of the security context token

A similar scenario except with Web Services Reliable Messaging (WS-ReliableMessaging) is possible from the WS-SecureConversation prospective. See the example for establishing a security context token to secure reliable messaging.

Flow for establishing a security context token to secure reliable messaging:

This example scenario includes functions that are required for the composite scenario of Web Services Reliable Messaging (WS-ReliableMessaging), WS-SecureConversation, and WS-Trust. The scenario describes how to use WS-SecureConversation with WS-ReliableMessaging, the scenario is described from the WS-SecureConversation perspective.

The flow of this Web Services Reliable Messaging (WS-ReliableMessaging) scenario is very similar to the flow of the WS-SecureConversation scenario, and the exchange of the application messages is very similar to the Secure Conversation scenarios. The main difference in the two example scenarios is that the WS-ReliableMessaging sequence is secured with the security context token and scopes the WS-ReliableMessaging sequence to the security context token.

The following figure describes a summary of the message flows that are required to establish a security context token to secure reliable messaging.

Figure 49. Messages exchange for the SCT and reliable messaging

Scenario

The WS-ReliableMessaging sequence is secured with the security context token and is scoping the WS-ReliableMessaging sequence to the security context token. This scenario focuses on the message exchanges that are using the security context token in the overall flow.

Note: The exact detail of how WS-ReliableMessaging is validating the WS-ReliableMessaging sequence, with respect to the security context token scoping, is not described.

Typically, to use secure conversation and a security context token to secure reliable messaging, the following steps are involved;

- The WS-ReliableMessaging run time calls APIs from the Web Services Security run time to get the UUID of the security context token for the session and also the API for serializing and deserializing the security context token for managed persistent for reliable recovery.

  Because of the security nature of the security context token, the WS-ReliableMessaging protocol makes sure that the serialized security context token in persistent store is protected.
- If there is already a security context token established the UUID of the existing security context token is returned to WS-ReliableMessaging. If there is no security context token already established, the Web Services Security run time initiates a call to the recipient to establish the security context token.

  The latter case is similar to the Secure Conversation scenario.
- After the WS-ReliableMessaging run time acquires the UUID of the security context token, the WS-ReliableMessaging run time scopes the CreateSequence message to the security context token by using the SecurityTokenReference (STR) argument in the CreateSequence message and responds with the CreateSequenceResponse message.

  The exchange of the application messages is very similar to the WS-SecureConversation scenario.
- The WS-ReliableMessaging run time responds with the CreateSequenceResponse message.

  The exchange of the messages is very similar to the exchange in the WS-SecureConversation scenario.
- The WS-ReliableMessaging run time sends a SequenceAcknowledgement message to acknowledge that the message is properly delivered and secured by the security context token.
- Finally, the WS-ReliableMessaging run time sends a TerminateSequence message to terminate the sequence and is secured by the security context token.

Enabling the distributed cache using synchronous update and token recovery:

To support secure conversation in a cluster environment, the distributed cache stores the shared state information. Version 7.0 and later of WebSphere Application Server uses MBeans to improve synchronous update of the cache across the cluster. In addition, persistent token support is provided by storing the token data in a database.

About this task

Synchronous update of shared information in the distributed cache is implemented in the product using an MBean solution. When update of the shared state information across cluster members is required, a synchronous blocking call is issued to replicate the token state changes to all the servers in the cluster. This solution removes the limitations of using session affinity for secure conversation in a cluster environment.

Perform the following high-level steps to enable distributed cache when using secure conversation for message-level protection in a cluster environment.

Procedure

1. In the administrative console for WebSphere Application Server, click **Services** > **Security cache**.
2. Click the check box to select the **Enable distributed cache** setting.
3. The distributed cache setting has three options. The first option is **Synchronous update of cluster members**. This option is selected by default, enabling the runtime to update all the cluster members with token information synchronously. If this is selected, then session affinity does not have to be enabled.

   The second option is **Asynchronous update of cluster members**, which you can select by clicking the corresponding radio button. For this option to work successfully, session affinity must be enabled. For information on enabling session affinity, read the topic Enable distributed cache and session affinity when using Secure Conversation. If Asynchronous update of cluster members is selected, skip steps 4 and 5.

   The third option is **Token recovery support**. To enable this option, click the corresponding radio button, then select a data source (database) from the **Cell level data sources** menu list. To create a data source, click the **Manage data sources** link. If Token recovery support is selected, skip steps 4 and 5.
4. This step is needed only if Synchronous update of cluster members is selected. Create a replication domain, as follows:
   a. In the administrative console, click **Environment > Replication domains > New**.
   b. Enter a name for the domain. For example, **ABCDomain**.
   c. In the Number of replicas section, click the radio button to select the **Entire Domain** option.
   d. Click **OK**, then **Save**, to save the modified configuration.
5. This step is needed only if Synchronous update of cluster members is selected. Enable the dynamic cache by performing the following steps for each server in the cluster:
   a. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Container Services** > **Dynamic cache service**.
   b. Select the **Enable cache replication** option.
   c. Select the replication domain name that you created in the previous step. For example, **ABCDomain**.
   d. Under Replication type. select **Both push and pull** from the menu list.
   e. Click **OK**, then click **Save** to save the modified configuration.

   Different clusters should use different replication domains. Likewise, cluster members from the same cluster should use the same replication domain. This ensures that synchronous update of cluster members performed by the Web Services Security runtime, and dynamic replication service updates of cluster members performed by the WebSphere Application Server dynamic cache runtime, are in sync.

Results

When the configuration steps are complete, you have enabled the distributed cache with either the default option, which is synchronous update of cluster members, or with asynchronous cluster update or with token recovery support. The token recovery support option uses a JDBC database to store the token state. This provides failover support for high availability of the token. If the server processing the request does not have access to the secure conversation token, the request fails, producing an error such as "null secure conversation token" or "invalid secure conversation token."

Configuring the token generator and token consumer to use a specific level of WS-SecureConversation:

Use the administrative console to configure the token generator or token consumer to use a specific level of the WS-SecureConversation OASIS specification standard. Select one of the two levels of token types supported: Secure Conversation Token v200502, or Secure Conversation Token v1.3.

About this task

WebSphere Application Server supports two levels of the OASIS standard for WS-SecureConversation including both the submission draft version (February 2005 draft specification) and version 1.3 of the standard, which was approved on March 1, 2007. Using the administrative console, configure the token generator so that the appropriate token type for a specific level of the standard is issued when a security token is requested.

Procedure

1. Log on to the administrative console and navigate to the panel where the token generator is configured by clicking **Services** > **Policy sets** > **General provider policy set bindings** or **General client policy set bindings**.
2. Click on the name of the binding you want to edit.
3. Click the **WS-Security** policy in the Policies table.
4. Click the **Authentication and protection** link in the Main message security policy bindings section.
5. Click **New token** to create a new token generator or consumer, or click an existing token link from the Protection Tokens table.
6. Enter a token name, then use the Token type drop-down menu to select a secure conversation token type.
   - To specify a submission draft token type, select **Secure Conversation Token v200502**.
   - To specify a version 1.3 token type, select **Secure Conversation Token v1.3**.
7. The local name is populated according to the token type you selected, as follows:
   - Local name for the submission draft token type: http://schemas.xmlsoap.org/ws/2005/02/sc/sct
   - Local name for the version 1.3 token type: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512

   The URI field is also filled in based on the token type.
8. Click to deselect the option **Tolerate Secure Conversation Token v200502** if you want to enforce use of only the version 1.3 tokens. This option specifies whether the provider should handle both Secure Conversation Token version 1.3 and Secure Conversation Token v200502. By default, the provider handles both versions.
9. Click **Apply** to create a secure conversation token of the selected type.

Web Services Secure Conversation standard:

Web Services Secure Conversation (WS-SecureConversation) is a proposed Organization for the Advancement of Structured Information Standards (OASIS) standard that defines mechanisms for establishing and sharing security contexts, and deriving keys from security contexts, to enable a secure conversation.

The base Web Services Security (WS-Security) standard from OASIS defines how to digitally sign and encrypt the SOAP message to provide message level protection. The standard also defines how to attach and reference a security token for digital signature and encryption. However, it does not provide session-based protection when a long series of related messages were exchanged. The WS-Security specification focuses on the *message authentication model*. This approach, while useful in many situations, could be subject to several forms of attack.

The WS-SecureConversation specification introduces the concept of a security context and its usage. The security context token is a new WS-Security token type that represents the security context abstract concept. The token is identified by a URI and consists of negotiated keys as well as other security related properties. The *context authentication model* authenticates a series of messages and, therefore, addresses

these concerns. The context authentication model increases the overall performance and security of the subsequent exchanges, but it requires additional communications when authentication happens prior to normal application exchanges.

Version 1.0 of the OASIS WS-SecureConversation specification defines extensions that build on the Web Services Security (WS-Security) and Web Services Trust (WS-Trust) standards to provide secure communication across one or more messages.

IBM, Microsoft, and other vendors have been working on the WS-SecureConversation specification since 2004. A draft of this document was jointly published in February, 2005. The WS-SecureConversation draft was submitted to the OASIS Web Service Secure Exchange Technical Committee (WS-SX TC), which was formed in December 2005, along with Web Services Trust (WS-Trust) and Web Services Security Policy (WS-SecurityPolicy) drafts in order to begin the standardization process.

A revised Version 1.1 draft version of the WS-SecureConversation specification standard was submitted to OASIS in February 2005 and further defines the extensions in Version 1.0. This specification defines extensions to allow security context establishment and sharing, and session key derivation. These extensions allow contexts to be established and potentially more efficient keys or new key material to be exchanged.

The most recent version of the specification standard is version 1.3, which was approved by the WS-SX TC on March 1, 2007. Key requirements in this level of the specification include derived keys and per-message keys, and extensible security contexts. WebSphere Application Server adds support for version 1.3 of WS-SecureConversation, providing improved error handling using the standard fault codes as defined in the specification.

The Web Services Secure Conversation (WS-SecureConversation) standard is a building block that is used in conjunction with the other web service and application-specific protocols such as Web Services Security and Web Services Trust to accommodate a wide variety of security models and technologies. WS-SecureConversation is built on top of the WS-Security and WS-Trust models to provide secure communication between services. The WS-SecureConversation draft specification describes how to establish a security context token between two parties, and the WS-Trust specification describes how to issue and exchange security tokens.

This WS-SecureConversation draft specification includes extensions to Web Services Security and:

- Describes the security context token.
- Defines how security contexts are established.
- Describes how security contexts are amended, renewed, and cancelled. Amending context is not supported by WebSphere Application Server.
- Specifies how derived keys are computed.
- Specifies how to associate a specific security context with an action, if multiple security contexts exist.

WebSphere Application Server supports the client establishing a secured conversation with the target service endpoint.

WebSphere Application Server supports the OASIS Version 1.1 submission draft, which became available in February 2005. The WebSphere Application Server does not support all of the functions in the submission draft. WebSphere Application Server support of WS-SecureConversation focuses on:

- A security context token that is established between the initiating party and the recipient party.
- The operations that are supported on security context token, such as Issue token, Renew token, and Cancel token.
- The derived key (both explicit and implied)

Secure conversation provided with WebSphere Application Server does not provide support for a security context token (SCT) that is acquired from a third-party trust server, and does not provide support for a security context token that is created by the client.

For information about WS-SecureConversation:

- See the IBM developerWorks website.
- See the schema for this specification: WS-SecureConversation schema
- Refer to the following namespace prefixes that are used for WS-SecureConversation: http://schemas.xmlsoap.org/ws/2005/02/sc and http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512

Trust service:

The security token service that is provided by WebSphere Application Server is called the trust service. The WebSphere Application Server trust service uses the secure messaging mechanisms of Web Services Trust (WS-Trust) to define additional extensions for the issuance, exchange, and validation of security tokens.

Web Services Trust (WS-Trust) is an OASIS standard that enables security token interoperability by defining a request/response protocol. This protocol allows SOAP actors, such as a web services client, to request of some trusted authority that a particular security token be exchanged for another.

WebSphere Application Server is not providing a full security token service that implements all the contents of the WS-Trust draft specification. The WebSphere Application Server support of WS-Trust focuses on establishing a security context token for secure conversation. WebSphere Application Server supports many of the security features described in version 1.3 of the WS-Trust OASIS standard, dated March 19, 2007.

Third party WS-Trust client

WebSphere Application Server does not provide a WS-Trust client implementation. You can choose to use a third-party WS-Trust-enabled client but, if you do, WebSphere Application Server does not support a third-party trust-enabled client. A trust client can facilitate the generation of these soap messages and the processing of the response, but the client is not required.

WebSphere Application Server focuses on the issuing, renewing, and canceling of the security context token for Web Services Secure Conversation (WS-SecureConversation).

The WS-Trust specification must be followed to make requests of the trust service. This specification includes the use of Web Services Addressing (WS-Addressing) headers. The WS-Addressing headers are specified in both the August 2004 or the August 2005 specifications. Per the specification, the SOAP body must consist of a single RequestSecurityToken (RST) element. This element can contain sub-elements as defined in the WS-Trust and WS-SecureConversation specifications.

You can secure the WS-Trust SOAP messages by using the bootstrap policy that is defined in the policy set. The bootstrap security policy is invoked in the process of an initiator establishing communication with an application service. Initial requests to services other than the application service are secured by using the bootstrap policy. These initial requests typically involve one or more requests to a security token service (STS), such as the WebSphere Application Server trust service. An example of a request might be acquiring the security context token necessary for WS-SecureConversation. An *initiator* is the role that initiates the original request and, in most cases, it is the client. The client bootstrap policy set must correspond to the trust service issue and renew attached policy sets for the endpoint. The trust service cancel and validate attached policy sets for the endpoint must correspond to the client's application policy set.

Websphere Application Server provides two ways to secure SOAP messages that are destined for the trust service. One way is to use the bootstrap policy that is defined in the policy set. A second way is to use the Web Services Security API (WSS API). Your application might use the WSS API to acquire the security context token for the programmatic, API-based WS-SecureConversation.

For Secure Conversation, a request from the client to an endpoint service is suspended while a new (second) request is generated and processed by the trust service. The security context token returned with the second request is used to derive keys that secure communications with the service.

High-level trust service functions

The following list includes WS-Trust-related functions that are currently supported in WebSphere Application Server. The list is not exhaustive and it focuses only on the high-level functions.

- The trust service component is embedded into and available on each WebSphere Application Server that processes the WS-Trust protocol messages.
- Communication is accomplished through the RequestSecurityToken (RST), RequestSecurityTokenCollection (RSTC), RequestSecurityTokenResponse (RSTR), and RequestSecurityTokenResponseCollection (RSTRC).

  Note: An RST request can be made to an external security token service (trust service). However, the restriction is that security context token, which is needed for WS-SecureConversation, must be provided by the WebSphere Application Server trust service.
- A security policy for each of the WS-Trust operations (issue, cancel, validate, and renew).
- Pre-configured Security Context Token provider which issues tokens for specific URL.
- Specification of a token provider's token-specific parameters (for example, expiration time).
- A security context token for WS-SecureConversation.
- Caching support for the security context token in both cluster and non-cluster environments. WebSphere Application Server issues security context tokens when requested if the request meets the security requirements. However, WS-SecureConversation provided by WebSphere Application Server only processes security context tokens that are issued by WebSphere Application Server.
- Note that WebSphere Application Server trust service only supports the security context token.
- Trust service supports both the Submission specification (2004/08) and the final specification (2005/08) versions of WS-Addressing.
- Trust service uses a default policy set called TrustServiceSecurityDefault, which includes WS-Security and WS-Addressing and provides default security for the issue and renew operations.
- Trust service uses a second default policy set called TrustServiceSymmetricDefault, which includes WS-Security and WS-Addressing and provides default security for the cancel and validate operations.

Trust service functions that are not supported

The following high-level WS-Trust functions that are not supported in WebSphere Application Server. The list is not exhaustive, and the list focuses only on key functions:

- No negotiation and exchange protocols are supported.
- No other token types are currently supported out of the box; only the security context token is supported.
- No Trust10 specifications from WS-SecurityPolicySet are supported.
- No unsolicited RequestSecurityTokenResponse (RSTR) is supported.
- A Request Security Token (RST) request cannot be issued to an external security token service (STS) to establish a secure conversation; only the embedded trust service is currently supported.
- Policy requests that are contained in the RST are not honored.
- The ability to amend a token (the amend operation) is not supported.

- A dedicated external endpoint for access to the token service is not supported; only the embedded trust service is currently supported.
- The trust services does not support the entropy element that contains an EncryptedKey.
- Delegation and forwarding are not supported.
- The OnBehalfOf element is not supported.
- The Key Exchange Token (KET) binding is not supported.

Trust service operations

WebSphere Application Server specifically supports the ability of the trust service, on behalf of the endpoint, to issue a security context token for WS-SecureConversation. The token-issuing support is currently limited only to the security context token. There is also trust policy management for defining a policy for the trust service to issue, cancel, validate, or renew tokens.

The token service supports the WS-Trust schema namespace. Within this namespace the following actions are supported:

- http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue
- http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Cancel
- http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Validate
- http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Renew

The token service also supports the WS-SecureConversation schema namespace. Within this namespace the following actions are supported:

- http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT
- http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Cancel
- http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT/Renew

An inbound RST for the security context token issue operation must contain an Entropy element. The Entropy element must contain a BinarySecret. The trust services does not support the Entropy element that contains an EncryptedKey.

Note that the trust service does not support unsolicited RSTR actions. In addition, the ability to amend a token is not supported by WebSphere Application Server. Also, see the section titled Trust service functions that are not supported.

Trust policy set-related files

The default trust service policy set for issue and renew is TrustServiceSecurityDefault. You can set up the corresponding policy set and binding for each service endpoint URL.

Security context token:

Web Services Trust (WS-Trust) and Web Services Secure Conversation (WS-SecureConversation) support in the application server provides the ability to issue a security context token (SCT). Requests for a security context token are processed by the security token service.

The security token service for WebSphere Application Server is called the trust service. However, the application server does not provide a full security token service that implements all the contents of the WS-Trust specification.

The secure session is referred to as *secure conversation* because the message protocols that are used are defined by WS-SecureConversation and WS-Trust. WebSphere Application Server supports secure conversation.

To request a security context token, a RequestSecurityToken (RST), which is defined by WS-Trust and WS-SecureConversation protocols, is sent to the service endpoint to which you are setting up a secure conversation. These requests are transparently rerouted to the trust service. The trust service processes the RST and responds with a RequestSecurityTokenResponse (RSTR). This response is returned to the requestor as if it was generated by the endpoint service.

The WebSphere Application Server token provider support is limited to the Security Context Token provider. WS-SecureConversation in the application server focuses on the establishing of the security context token between the initiating party and the recipient party for secure conversation.

WebSphere Application Server includes caching support for the Security Context Token in both cluster and non-cluster environments as well as on both the client and server. WebSphere Application Server also provides trust policy set management for each of the trust service operations: issue, cancel, validate, and renew. Trust system policy sets can be managed for each of these trust operations relative to an explicit service endpoint or the trust service default. The default trust service policy set for a trust operation is enforced when there is not an explicit attachment.

See the information about Web Services Trust for the WS-Trust functions that are supported.

For the security context token, you can:

- Configure the security context token provider for WS-SecureConversation, providing issue, renew and cancel operations.
- Configure the trust service to issue a security context token for access to a specific endpoint service (target).
- Configure the security requirements for access to the trust service and applications. WebSphere Application Server provides pre-configured application policy sets and trust service policy sets to assist with this configuration.
- Define a system policy for each of the four trust service operations: issue, cancel, validate, and renew. These policies are configured for the default or a specific endpoint service. Note that the amend operation is not supported.

The Security Context Token provider does not support the following operations:

- WS-SecureConversation amend
- Negotation to establish Secure Conversation
- WS-Trust key exchange requests
- Client-initiated RequestSecurityTokenResponse (RSTR) and RequestSecurityTokenResponseCollection (RSTRC) requests
- WS-SecurityPolicy trust assertions

Definitions

To better understand security tokens, the following terms are defined:

security token
A security token represents a collection of claims.

security context
A security context is an abstract concept that refers to an established authentication state and negotiated key or keys that can have additional security-related properties. A security context needs to be created and shared by the communicating parties before being used. A security context is shared among the communicating parties for the lifetime of a communications session and a security context token is the wire representation of this abstract security context.

WebSphere Application Server does not support a security context token created by one of the communicating parties and propagated with a messageWebSphere Application Server does not support creating a security context token through negotiation and exchanges.

security context token
A security context token is a wire representation of that security context abstract concept, which allows a context to be named by a URI and to be used with Web Services Security. A secured communication with a security context token between two parties is realized with WS-Trust and WS-SecureConversation.

security token service
A security token service (STS) is a web service that issues security tokens, meaning it makes assertions that are based on evidence that it trusts, to whoever trusts it (or to specific recipients).

Trust service
The trust service is the security token service and supporting code that is provided by Websphere Application Server.

RequestSecurityToken (RST)
A RST is a message sent to a security token service to request a security token.

RequestSecurityToken Response (RSTR)
A RSTR is a response to a request for a security token from a security token service to a requestor after receiving an RST message.

To communicate trust, a service requires proof, such as a signature, to prove knowledge of a security token or set of security tokens. A service itself can generate tokens or it can rely on a separate security token service to issue a security token with its own trust statement. Note that, for some security token formats, communicating trust can just be a re-issuance or a co-signature that forms the basis of trust brokering.

Syntax for the <wsc:SecurityContextToken> element

A security context is shared among the communicating parties for the lifetime of a communications session and a security context token is the wire representation of this abstract security context.

In the WS-SecureConversation specification, a security context is represented by the <wsc:SecurityContextToken> security token. The following URI represents the security context token type that is required to establish a secure conversation.

```
http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
```

The syntax for <wsc:SecurityContextToken> element is as follows:

```
<wsc:SecurityContextToken wsu:Id="..." ...>
    <wsc:Identifier>...</wsc:Identifier>
    <wsc:Instance>...</wsc:Instance>
    ...
</wsc:SecurityContextToken>
```

The security context token does not support references to it by using key identifiers or key names. All references must use an ID (to a *wsu:Id* attribute) or use a URI reference, <wsse:Reference>, to the <wsc:Identifier> element in the security context token.

RST and RSTR examples to issue a security token

This example shows a RST request to issue a security token. The URI `http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct`, which is used in this example, represents the token type:

```
<wsc:SecurityContextToken>
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

   <soapenv:Header>
```

```
 <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing"
  soapenv:mustUnderstand="0">
  http://localhost:80/WSSampleSei/EchoService
 </wsa:To>
 <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing"
  soapenv:mustUnderstand="0">
  fc0632828e1252b4:487cee53:11cbfa7916e:-7fb6
 </wsa:MessageID>
 <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing"
  soapenv:mustUnderstand="0">
  http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT
 </wsa:Action>
</soapenv:Header>

<soapenv:Body>

 <wst:RequestSecurityToken
  xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
  Context="http://www.ibm.com/login/">
  <wst:TokenType>
   http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
  </wst:TokenType>
  <wst:RequestType>
   http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
  </wst:RequestType>
  <wsp:AppliesTo
   xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
   -
   <wsa:EndpointReference
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
    <wsa:Address>
     http://localhost:80/WSSampleSei/EchoService
    </wsa:Address>
   </wsa:EndpointReference>
  </wsp:AppliesTo>
  <wst:Entropy>
   <wst:BinarySecret
    Type="http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce">
    zb//KsawV6DmfC8kB6vNOQ==
   </wst:BinarySecret>
  </wst:Entropy>
  <wst:KeySize>128</wst:KeySize>
 </wst:RequestSecurityToken>
</soapenv:Body>
</soapenv:Envelope>
```

This example shows a RSTR request to issue a security token:

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

 <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Action>
   http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/SCT
  </wsa:Action>
  <wsa:RelatesTo>
   fc0632828e1252b4:487cee53:11cbfa7916e:-7fb6
  </wsa:RelatesTo>
 </soapenv:Header>

 <soapenv:Body>
  <wst:RequestSecurityTokenResponseCollection
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
   <wst:RequestSecurityTokenResponse
    Context="http://www.ibm.com/login/">
    <wst:RequestedSecurityToken>
     <wsc:SecurityContextToken
      xmlns:wsc="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
      wsu:Id="uuid:FFA51A32EB818FB6EA1222986227363">
      <wsc:Identifier>
       uuid:FFA51A32EB818FB6EA1222986227346
      </wsc:Identifier>
      <wsc:Instance>
       uuid:FFA51A32EB818FB6EA1222986227345
      </wsc:Instance>
     </wsc:SecurityContextToken>
    </wst:RequestedSecurityToken>
```

```
    <wsp:AppliesTo
     xmlns:wsp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
     <wsa:EndpointReference
      xmlns:wsa="http://www.w3.org/2005/08/addressing">
      <wsa:Address>
       http://localhost:80/WSSampleSei/EchoService
      </wsa:Address>
     </wsa:EndpointReference>
    </wsp:AppliesTo>
    <wst:RequestedProofToken>
     <wst:ComputedKey>
      http://docs.oasis-open.org/ws-sx/ws-trust/200512/CK/PSHA1
     </wst:ComputedKey>
    </wst:RequestedProofToken>
    <wst:Entropy>
     <wst:BinarySecret
      Type="http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce">
      rF1Yp5zhRhamLQNPAOm4TA==
     </wst:BinarySecret>
    </wst:Entropy>
    <wst:Lifetime>
     <wsu:Created
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      2008-10-02T22:23:44.765Z
     </wsu:Created>
     <wsu:Expires
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
      2008-10-02T22:35:44.765Z
     </wsu:Expires>
    </wst:Lifetime>
    <wst:RequestedAttachedReference>
     <wsse:SecurityTokenReference
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:Reference
       URI="#uuid:FFA51A32EB818FB6EA1222986227363"
       ValueType="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct" />
     </wsse:SecurityTokenReference>
    </wst:RequestedAttachedReference>
    <wst:RequestedUnattachedReference>
     <wsse:SecurityTokenReference
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <wsse:Reference
       URI="uuid:FFA51A32EB818FB6EA1222986227346"
       ValueType="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct" />
     </wsse:SecurityTokenReference>
    </wst:RequestedUnattachedReference>
    <wst:Renewing Allow="true" OK="false" />
    <wst:KeySize>128</wst:KeySize>
   </wst:RequestSecurityTokenResponse>
  </wst:RequestSecurityTokenResponseCollection>
 </soapenv:Body>
</soapenv:Envelope>
```

RST and RSTR examples to cancel a security token

This example shows a RST request to cancel a security token.

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

 <soapenv:Header>
  <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   http://localhost:80/WSSampleSei/EchoService
  </wsa:To>
  <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   fc0632828e1252b4:-270287b7:11cc22c16ed:-7fa8
  </wsa:MessageID>
  <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Cancel
  </wsa:Action>
 </soapenv:Header>

 <soapenv:Body>
  <wst:RequestSecurityToken
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
```

```
   Context="http://www.ibm.com/login/">
   <wst:TokenType>
    http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
   </wst:TokenType>
   <wst:RequestType>
    http://docs.oasis-open.org/ws-sx/ws-trust/200512/Cancel
   </wst:RequestType>
   <wsp:AppliesTo
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <wsa:EndpointReference
     xmlns:wsa="http://www.w3.org/2005/08/addressing">
     <wsa:Address>
      http://localhost:80/WSSampleSei/EchoService
     </wsa:Address>
    </wsa:EndpointReference>
   </wsp:AppliesTo>
   <wst:CancelTarget>
    <wsc:SecurityContextToken
     xmlns:wsc="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
     wsu:Id="uuid:AC4764EB4BE91011501223028453769">
     <wsc:Identifier>
      uuid:AC4764EB4BE91011501223028453768
     </wsc:Identifier>
     <wsc:Instance>
      uuid:AC4764EB4BE91011501223028453751
     </wsc:Instance>
    </wsc:SecurityContextToken>
   </wst:CancelTarget>
  </wst:RequestSecurityToken>
 </soapenv:Body>
</soapenv:Envelope>
```

This example shows a RSTR request to cancel a security token:

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:wsa="http://www.w3.org/2005/08/addressing">

 <soapenv:Header>
        <wsa:Action>
          http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Cancel
      </wsa:Action>
      <wsa:RelatesTo>
          fc0632828e1252b4:-270287b7:11cc22c16ed:-7fa8
      </wsa:RelatesTo>
   </soapenv:Header>

  <soapenv:Body>
      <wst:RequestSecurityTokenResponse
           Context="http://www.ibm.com/login/"
                xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
         <wst:RequestedTokenCancelled>
      </wst:RequestSecurityTokenResponse>
   </soapenv:Body>
</soapenv:Envelope>
```

RST and RSTR examples to renew a security token

This example shows a RST request to renew a security token.

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

 <soapenv:Header>
  <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   http://localhost:80/WSSampleSei/EchoService
  </wsa:To>
  <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   fc0632828e1252b4:487cee53:11cbfa7916e:-7f8e
  </wsa:MessageID>
  <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Renew
  </wsa:Action>
```

```
 </soapenv:Header>

  <soapenv:Body>
  <wst:RequestSecurityToken
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
   Context="http://www.ibm.com/login/">
   <wst:TokenType>
    http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
   </wst:TokenType>
   <wst:RequestType>
    http://docs.oasis-open.org/ws-sx/ws-trust/200512/Renew
   </wst:RequestType>
   <wst:RenewTarget>
    <wsc:SecurityContextToken
     xmlns:wsc="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
     wsu:Id="uuid:FFA51A32EB818FB6EA1223026418869">
     <wsc:Identifier>
      uuid:FFA51A32EB818FB6EA1223026418868
     </wsc:Identifier>
     <wsc:Instance>
      uuid:FFA51A32EB818FB6EA1223026418867
     </wsc:Instance>
    </wsc:SecurityContextToken>
   </wst:RenewTarget>
   <wsp:AppliesTo
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <wsa:EndpointReference
     xmlns:wsa="http://www.w3.org/2005/08/addressing">
     <wsa:Address>
      http://localhost:80/WSSampleSei/EchoService
     </wsa:Address>
    </wsa:EndpointReference>
   </wsp:AppliesTo>
   <wst:Entropy>
    <wst:BinarySecret
     Type="http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce">
     U8rH9l/wLV1gpsBf/yCooA==
    </wst:BinarySecret>
   </wst:Entropy>
   <wst:KeySize>128</wst:KeySize>
  </wst:RequestSecurityToken>
 </soapenv:Body>
</soapenv:Envelope>
```

This example shows a RSTR request to renew a security token:

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

 <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Action>
   http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/RenewFinal
  </wsa:Action>
  <wsa:RelatesTo>
   fc0632828e1252b4:487cee53:11cbfa7916e:-7f8e
  </wsa:RelatesTo>
 </soapenv:Header>

 <soapenv:Body>
  <wst:RequestSecurityTokenResponse
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
   Context="http://www.ibm.com/login/">
   <wst:RequestedSecurityToken>
    <wsc:SecurityContextToken
     xmlns:wsc="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
     wsu:Id="uuid:FFA51A32EB818FB6EA1223026990448">
     <wsc:Identifier>
      uuid:FFA51A32EB818FB6EA1223026418868
     </wsc:Identifier>
     <wsc:Instance>
      uuid:FFA51A32EB818FB6EA1223026990447
     </wsc:Instance>
    </wsc:SecurityContextToken>
   </wst:RequestedSecurityToken>
   <wst:Entropy>
    <wst:BinarySecret
```

```
     Type="http://docs.oasis-open.org/ws-sx/ws-trust/200512/Nonce">
     lFkKSI/pajtTZzRpQalNMA==
    </wst:BinarySecret>
   </wst:Entropy>
   <wst:Lifetime>
    <wsu:Created
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
     2008-10-03T09:43:07.421Z
    </wsu:Created>
    <wsu:Expires
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
     2008-10-03T09:55:07.421Z
    </wsu:Expires>
   </wst:Lifetime>
   <wst:RequestedAttachedReference>
    <wsse:SecurityTokenReference
     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
     <wsse:Reference
      URI="#uuid:FFA51A32EB818FB6EA1223026990448"
      ValueType="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct">
     </wsse:Reference>
    </wsse:SecurityTokenReference>
   </wst:RequestedAttachedReference>
   <wst:Renewing Allow="true" OK="false"></wst:Renewing>
  </wst:RequestSecurityTokenResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

RST and RSTR examples to validate a security token

This example shows a RST request to validate a security token.

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

 <soapenv:Header>
  <wsa:To xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   http://localhost:80/WSSampleSei/EchoService
  </wsa:To>
  <wsa:MessageID xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   fc0632828e1252b4:-673f2c18:11cc328886a:-7fa7
  </wsa:MessageID>
  <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing"
   soapenv:mustUnderstand="0">
   http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Validate
  </wsa:Action>
 </soapenv:Header>

  <soapenv:Body>
  <wst:RequestSecurityToken
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
   Context="http://www.ibm.com/login/">
   <wst:TokenType>
    http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
   </wst:TokenType>
   <wst:RequestType>
    http://docs.oasis-open.org/ws-sx/ws-trust/200512/Validate
   </wst:RequestType>
   <wst:ValidateTarget>
    <wsc:SecurityContextToken
     xmlns:wsc="http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512"
     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
     wsu:Id="uuid:6B77A2DA28C1E523BD1223045150688">
     <wsc:Identifier>
      uuid:6B77A2DA28C1E523BD1223045150687
     </wsc:Identifier>
     <wsc:Instance>
      uuid:6B77A2DA28C1E523BD1223045150670
     </wsc:Instance>
    </wsc:SecurityContextToken>
   </wst:ValidateTarget>
   <wsp:AppliesTo
    xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <wsa:EndpointReference
     xmlns:wsa="http://www.w3.org/2005/08/addressing">
     <wsa:Address>
```

```
      http://localhost:80/WSSampleSei/EchoService
     </wsa:Address>
    </wsa:EndpointReference>
   </wsp:AppliesTo>
  </wst:RequestSecurityToken>
 </soapenv:Body>
</soapenv:Envelope>
```

This example shows a RSTR request to validate a security token:

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">

 <soapenv:Header xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <wsa:Action>
   http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/ValidateFinal
  </wsa:Action>
  <wsa:RelatesTo>
   fc0632828e1252b4:-673f2c18:11cc328886a:-7fa7
  </wsa:RelatesTo>
 </soapenv:Header>

 <soapenv:Body>
  <wst:RequestSecurityTokenResponse
   xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
   Context="http://www.ibm.com/login/">
   <wst:Status>
    <wst:Code>
     http://docs.oasis-open.org/ws-sx/ws-trust/200512/status/valid
    </wst:Code>
   </wst:Status>
  </wst:RequestSecurityTokenResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

For additional information, review the two example scenario topics that discuss establishing the security context token.

System policy sets:

A policy set is a named collection of Quality of Service (QoS) policies. You can use either the administrative console or the wsadmin commands to manage system policy sets. Policy sets can be created, deleted, copied, imported or exported.

A policy set can be shared by multiple resources, such as applications, services, inbound or outbound service endpoints, and operations. Default policy sets are installed using profile augmentation. A policy set can also be imported. A policy set does not have its own bindings. You must attach a policy set to a resource, and then assign a binding to the attachment.

Note: When attempting to connect to a web service from a thin client, verify that the resources that you are specifying are valid before running the updatePolicySetAttachment command. No configuration changes are made if the requested resource does not match a resource in the attachment file for the application.

A client application can dynamically select a policy suite (reference by name from an application-level policy suites list). Options shown in the administrative console list are based on the type of template that is selected to create the policy set. For example, the SecureConversation policy type is made up of policies for both WSSecurity and WSAddressing.

There are two types of policy sets:

- Application policy sets
- System/trust policy sets

WebSphere Application Server provides predefined system policy sets. For example, WebSphere Application Server provides the following system policy sets by default for the security trust service:

- TrustServiceSecurityDefault

  This trust policy set specifies the asymmetric algorithm as well as the public and private keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using RSA. Message confidentiality is provided by encrypting the body and signature using RSA. This policy set follows the WS-Security specifications for the issue and renew trust operation requests.
- TrustServiceSymmetricDefault

  This policy set specifies the symmetric algorithm as well as the derived keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using HMAC-SHA1. Message confidentiality is provided by encrypting the body and signature using AES. This policy set follows the WS-Security and Secure Conversation specifications for validate and cancel trust operation requests.
- SystemWSSecurityDefault

  This policy set specifies the asymmetric algorithm and both the public and private keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using RSA encryption. Message confidentiality is provided by encrypting the body and signature using RSA encryption

You cannot edit default system policy sets. However, you can create your own custom system policy set, which can be edited later. Copy or export a default or existing custom system policy set to create the new custom policy set. System policy sets can also be imported from a predefined location, or from the default repository. Add one or more policies to each policy set. For example, add any of the following existing policies:

- Custom properties
- HTTP transport
- JMS transport
- SSL Transport
- WS-Addressing
- WS-Security

The HTTP transport policy can be used for HTTPS, basic authorization, compression, and binary encoding transport methods.

Web Services Trust standard:

Web Services Trust (WS-Trust) is a proposed Organization for the Advancement of Structured Information Standards (OASIS) standard that enables security token interoperability by defining a request/response protocol. This protocol allows SOAP actors, such as a web services client, to request of some trusted authority that a particular security token be exchanged for another. The trust service, which is provided with WebSphere Application Service, uses the secure messaging mechanisms of WS-Trust to define additional extensions for the issuance, exchange, and validation of security tokens.

WS-Trust defines a request and response protocol for security token exchange. A client sends a RequestSecurityToken (RST) to a security token service. The request includes the security token that the client is asking to be exchanged. The security token service responds back with a RequestSecurityTokenResponse (RSTR) that contains the new token.

In addition to the token exchange, the WS-Trust request/response protocol is general enough to support token *issuance*, where the client presents a claim to the trust service for the service to authorize through the issuance of a corresponding security token. Token *validation* is where the client presents a token to the trust service and asks that its validity be determined.

Also, WS-Trust enables the issuance and dissemination of credentials within different trust domains. To secure a communication between two parties, the two parties must exchange security credentials (either directly or indirectly). Each party must first determine if they can trust the asserted credentials of the other party.

The OASIS WS-Trust specification defines extensions to Web Services Security (WS-Security) for issuing and exchanging security tokens and for providing ways to establish and access the presence of trust relationships. Using these extensions, applications can engage in secure communication, and these extensions are designed to work with the general web services framework. The general web services framework includes the WSDL service descriptions, UDDI businessServices and bindingTemplates, and SOAP messages.

The WebSphere Application Server support of WS-Trust focuses on establishing a security context token for Web Services Secure Conversation (WS-SecureConversation). The WS-Trust support focuses on the four actions for the security context token: issue, renew, validate, and cancel. Also supported for WS-Trust Version 1.3 are collection requests for the same actions: issue, renew, validate and cancel. The major component for WS-Trust that WebSphere Application Server supports is the security token service, which is referred to as the trust service.

Support for submission draft and approved levels of the WS-Trust standard

Version 6.1 and later of WebSphere Application Server supports the WS-Trust 2005 Submission Draft specification (Version 1.1). However, WebSphere Application Server does not provide a full security token service that implements all the contents of the WS-Trust draft specification.

Support for the approved version 1.3 specification, which is dated March 2007, is provided for WebSphere Application Server version 7.0 and later. The Security Context Token (SCT) provider supports the OASIS version 1.3 specifications for WS-Trust and WS-SecureConversation. There is a configuration option that allows support for the two different levels of the WS-Trust standard to co-exist on the same server. This provides interoperability between systems and products that support different specification levels. See the topic Configuring the security context token provider for the trust service using the administrative console for details.

A setting is also provided to specifically disable support for the WS-Trust 2005 Submission Draft specification (Version 1.1) for the Security Context Token provider. For more information about this property, refer to the topic Disabling the draft standard level for the Security Context Token.

Processing a trust service request depends on the specifications referenced in the request. Also, the trust service response is determined by the level of the specification used in the request.

For more information about WS-Trust:

- See the IBM developerWorks website.
- See the schema for the specification: http://docs.oasis-open.org/ws-sx/ws-trust/200512
- Refer to the `wst` namespace prefix that is used for WS-Trust in the Web Services Trust Language (WS-Trust) specification dated March 2007.

Configuring system policy sets using the administrative console:

By defining a custom policy set or defining assertions about how services are defined, you can configure Web Services Security. You can use the administrative console to manage custom policy sets.

Before you begin

A policy set specifies a set of common message policy assertions that can be specified within a policy. For example, a policy set can define general security policy assertions that apply to other protocols, such as Web Services Security (WS-Security), SOAP messages, Web Services Secure Conversation (WS-Secure Conversation) and Web Services Trust (WS-Trust).

There are two main types of policy sets; application policy sets and system policy sets. Application policy sets are used for business-related assertions. These assertions are related to the business operations that are defined in the Web Services Description Language (WSDL) file. System policy sets, on the other hand, are used for non-business-related system messages. These messages are defined in other specifications which apply qualities of service (QoS). Examples of QoS are the request security token (RST) messages that are defined in WS-Trust, the create sequence messages that are defined in WS-Reliable Messaging, and the metadata exchange messages defined by WS-MetadataExchange.

Important: Use system policy sets with the trust service, or Web Services MetadataExhange (WS-MEX). The requestor (client) must utilize Java API for XML-Based Web Services (JAX-WS) only. Requestors which use Java API for XML-based remote procedure calls (JAX-RPC) are incompatible with the policy set QOS.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

About this task

Only custom policy sets can be modified. Default system policy sets are read only and cannot be changed.

Procedure

1. To define system policy sets, click **Services** > **Policy sets** > **System policy sets**.
2. Click one of the following actions to work with the system policy set configurations:

   New To create a system policy set configuration. Enter a unique name for the system policy set configuration in the Name field. For example, you might specify `EcommerceTrustServiceSecurity`.

   Delete To delete an existing configuration. Select the check box next to an existing policy set name, and click **Delete**.

   Copy To copy an existing configuration. Select the check box next to an existing policy set name, and click **Copy**.

   Import
   To import an existing configuration. Select the check box next to an existing policy set name, and click **Import**. For more information, read about importing policy sets using the administrative console.

   Export
   To export an existing configuration. Select the check box next to an existing policy set name, and click **Export**. For more information, read about exporting policy sets using the administrative console.
3. To edit the settings of an existing policy set configuration, click the link for the existing custom system policy set that you want to change. Use the administrative console to modify existing custom policy sets that have been created.
4. Optional: If creating a policy set, enter a short description for the new policy set. Default policy sets can only be viewed. For a custom policy set, edit the brief description of the policy set in the

Description field. This description displays in the list on the System policy sets panel. The description should be meaningful to you and other potential users of this policy set.

5. If creating a new policy set, click **Apply**. The policy set name must be applied before you can add policy types to the new policy set.
6. Optional: If needed, add the policy type information, or change the policy types for an existing system policy set. You can add, delete, enable, or disable policy types for the selected policy set. You can add any valid policy types to the policy set collection. The following are available policy types for system policy sets:
   - HTTP transport - for HTTP transport policies
   - SSL transport - for HTTPS transport policies
   - WS-Addressing - for endpoint addressing policies
   - WS-Security - for secure SOAP messages policies
7. Click **OK** and then click **Save** to save the information directly to the master configuration.

Results

You have provided the basic information to create a system policy set. You can also create a new or update an existing system policy set for the WebSphere Application Server trust service, or Web Services MetadataExhange (WS-MEX), using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

What to do next

After creating a system policy set and adding the policy types, attach the system policy set to a trust service operation for an endpoint, or attach it to one of the trust service default operations.

Defining a new system policy set using the administrative console:

Use policy sets, or assertions, to define system service operations, for your Web Services Security configuration. Whenever you create a new policy set, you must add policy types to the policy set. You can add HTTP Transport, WS-Addressing, WS-Security, and SSL Transport policy types to the system policy set collection.

Before you begin

A policy set specifies a set of common message policy assertions that can be specified within a policy. For example, a policy set can define general security policy assertions that apply to other protocols such as Web Services Security (WS-Security), SOAP messages, Web Services Trust (WS-Trust), and Web Services Secure Conversation (WS-SecureConversation).

Important: Use system policy sets with the trust service only. The requestor (client) must utilize Java API for XML-Based Web Services (JAX-WS) only. Requestors which use Java API for XML-based remote procedure calls (JAX-RPC) are incompatible with the policy set QOS.

About this task

Use the system policy sets to configure access to the WebSphere Application Server trust service. You can create and define a custom system policy set.

Procedure

1. Using the administrative console, click **Services > Policy sets > System policy sets** .
2. To create a system policy set and add a policy type, click **New**.
3. Enter a name for the policy set in the **Name** field. The name must be unique for the new system policy set. For example: EcommerceTrustServiceSecurity

4. Enter a brief description of the policy set in the **Description** field. This description displays in the System Policy Sets collection. The description should be descriptive enough for you and other potential users to identify the policy set.
5. Click **Apply** to apply the name and description information.
6. Click **Add** to add a trust policy by selecting one from the policies listed. The following policies are available to use for system policy sets:
   - HTTP transport - for HTTP transport policies
   - SSL transport - for HTTPS transport policies
   - WS-Addressing - for endpoint addressing policies
   - WS-Security - for secure SOAP messages policies
7. Click **Save** to save directly to the master configuration.

Results

You have provided the basic information to create or modify a policy set. You can also create a new or update an existing policy set for the WebSphere Application Server trust service using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

What to do next

After creating or modifying a system policy set and adding the policy types, attach the policy set to an endpoint operation or attach it to one of the trust service default operations.

System policy set collection:

Use this panel to create and manage policy sets. A policy set is a named collection of policies. System policy sets, or assertions about how services are defined, are used to configure access to the trust service.

There are two main types of policy sets; application policy sets and system policy sets. Application policy sets are used for business-related assertions. These assertions are related to the business operations that are defined in the Web Services Description Language (WSDL) file. System policy sets, on the other hand, are used for non-business-related system messages. These messages are defined in other specifications which apply qualities of service (QoS). Examples of QoS are the request security token (RST) messages that are defined in WS-Trust, the create sequence messages that are defined in WS-Reliable Messaging, and the metadata exchange messages defined by WS-MetadataExchange.

To view this administrative console page, click **Services** > **Policy sets** > **System policy sets**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Select:

Provides a check box next to the name of an existing system policy set that you want to select for further actions.

To manage existing system policy sets, select the check box for a system policy set and then select one of the following actions:

Actions	Description
Delete	Removes one or more selected system policy sets.

Actions	Description
Copy	Opens a new panel where you can create a copy of the selected existing policy set. Provide a unique name and, optionally, a description for the copied policy set. You must also specify whether to transfer the attachment and binding from the original version to the copy. You can select only one policy set to be copied at one time.
Import	Imports a policy set. This is a menu item with the option of importing a policy set from a default repository or a selected location. You can select and import the default policy sets from the default repository. The default repository for the import function in the administrative console is the directory which contains the default policy sets. The administrative console also displays the default policy sets in a list which includes descriptions, to allow you to select the desired policy set that you want to import.
Export	Opens a new panel where you can export the selected policy set. You can select only one policy set to be exported at one time.

New:

Specifies to create and define a custom system policy set.

Name:

Provides a list of available system policy sets.

This column displays a list of default and custom system policy set names. WebSphere Application Server provides several default system policy sets:

- **TrustServiceSecurityDefault** is a default trust policy set. This trust policy set specifies the asymmetric algorithm as well as the public and private keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using RSA. Message confidentiality is provided by encrypting the body and signature using RSA. This policy set follows the WS-Security specifications for the issue and renew trust operation requests.
- **TrustServiceSymmetricDefault** is a default trust policy set. This trust policy set specifies the symmetric algorithm as well as the derived key algorithms to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using HMAC-SHA1. Message confidentiality is provided by encrypting the body and signature using AES. This policy set follows the WS-Security and WS-SecureConversation specifications for the validate and cancel trust operation requests.
- **SystemWSSecurityDefault** is a default system policy set that specifies the asymmetric algorithm and both the public and private keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using RSA encryption. Message confidentiality is provided by encrypting the body and signature using RSA encryption.

All custom system policy sets are also displayed in the list. Click the system policy set name to view additional details about the selected policy set.

Data type:	String
Defaults:	TrustServiceSecurityDefault, TrustServiceSymmetricDefault or SystemWSSecurityDefault

Editable:

Provides information as to whether the system policy set can be edited.

This column shows whether the policy set is a user-defined, custom policy set that can be edited or whether the policy set is a default policy set that is not editable. Values displayed in this field are: `Editable` or `Not editable`. You can change the properties for a default, not editable policy set by copying it, and then modifying the properties of the copy. For more information, read about copying default policy set and bindings settings.

Important: Even though a policy set is identified as not editable, it is deletable. For example, you cannot edit information for the default system policy set, but you can delete the policy set.

Data type: String
Default: Not editable

Description:

Provides brief descriptions of the system policy sets that currently exist.

This column provides a brief description of the policy sets that are available. You cannot edit information for the default system policy sets. For custom policy sets that you create, you can create the description when you create the policy set. Or, you can edit any custom policy set and modify the description on the details panel at any time. The description field is optional.

System policy set settings:

Use this panel to create a new system policy set or to edit information about an existing custom system policy set. System policy sets, or assertions about how services are defined, are used to configure access to the trust service.

To view this administrative console page, complete one of the following procedures:

- **Services** > **Policy sets** > **System policy sets** > ***policyset_name***
- **Services** > **Policy sets** > **System policy sets** > **New**

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Important: You can edit the fields on this page only if the policy set is a custom trust policy set. You cannot edit default trust policy sets.

Name:

Specifies the name of the trust policy set.

This field displays the name of the existing custom policy set that you selected. If a new policy set is being created, this field is blank. Enter a policy set name.

Data type: String

Description:

Specifies a brief description of the new or existing custom policy set.

This field provides a brief description for the existing policy set that is displayed in the Name field. If a new policy set is being created, this field is blank. Enter a brief policy set description to help distinguish it from other policy sets. You cannot change the descriptions of the default policy sets.

Data type: String

Policies:

Specifies a collection of trust-related policies.

The Policies section displays a list of pre-configured system-level trust policies. If the system policy set is a default trust policy set, policies cannot be added, deleted, enabled, or disabled. If the system policy set is an existing custom trust policy set, you can change the policies. If you are creating a new custom trust policy set, you can add new policies. You can also delete, enable, or disable any policies that are added.

Select:

Specifies that you want to select an existing policy for further actions.

Click `Add` to display a list of valid policies that you can add to the named trust policy set.

To manage existing system policies, select the check box for a policy and select one of the following actions:

Actions	Description
Delete	Removes one or more policies from the named custom policy set.
Enable	Specifies that the policy is enabled for the policy set and displays `Enabled` in the State column.
Disable	Specifies that the policy is disabled for the policy set. The policy remains in the list but displays `Disabled` in the State column.

Policy:

Specifies the name of the policy.

This column displays one or more of these trust policies:

- **Custom properties** – for custom property policies
- **HTTP transport** – for HTTP transport policies
- **SSL transport** – for HTTPS transport policies
- **WS-Addressing** – for endpoint addressing policies
- **WS-Security** – for secure SOAP messages policies

State:

Specifies an enabled or disabled state for each policy.

This column displays whether the policy is enabled or disabled for each of the policies that are listed in the Policy column. For example, to change the state of the policy from enabled to disabled, select the check box for the policy, and click **Disable**.

Description:

Displays a brief description of the policy.

You can view these descriptions only.

Configuring attachments for the trust service using the administrative console:

You can attach the trust service operations for a service endpoint to a system policy set and binding. Each new endpoint that is specified initially has the following four operations: issue, renew, cancel, and validate. By default, all endpoints inherit the policy set and binding that are attached to the respective trust service operation under Trust Service Defaults. However, you can explicitly attach a different policy set.

Before you begin

First you must define your policy sets and bindings. *Policies* describe the protection or quality of service that is provided (such as message security, transport and so forth). *Bindings* specify some details about how to implement the policy, such as: the path for the keystore file, the class name of the token generator, or the JAAS configuration name.

Important: Use system policy sets with the trust service only. The requestor (client) must utilize Java API for XML-Based Web Services (JAX-WS) only. Requestors which use Java API for XML-based remote procedure calls (JAX-RPC) are incompatible with the policy set QOS.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

About this task

You can attach the trust service operations for a new endpoint to an existing policy set and binding. For each new service endpoint that is specified, four trust service operations (cancel, renew, validate and issue) change from having inherited attachments to being explicitly attached. The four operations are attached to the respective policy set and binding as specified in Trust Service Defaults. Then you can change the attachment to the desired existing policy set and binding.

An endpoint policy set consists of two sections: a bootstrap section and an application section. The system policy set attached to the Issue and renew trust service operations for a specific endpoint must correspond to the bootstrap section of the policy set for that endpoint. The system policy set attached to the Cancel and Validate trust service operations for a specific endpoint must correspond to the application section of the policy set for that endpoint.

This task describes how to manage trust service operations for service endpoint URLs that you want to attach to a system policy set and binding. To complete the configuration of the WebSphere Application Server trust service, you must also complete the following task:

- Create or manage targets. You can create explicit assignments for new service endpoints (targets) or manage endpoints that have a security token explicitly assigned or that inherit the Trust Service Default token.

The sample general bindings that are provided with the product are initially set as the global security (cell) default bindings. The default service provider binding and the default service client bindings are used when no application specific bindings or trust service bindings are assigned to a policy set attachment. For trust service attachments, the default bindings are used when no trust specific bindings are assigned. If you do not want to use the provided Provider sample as the default service provider binding, you can select an existing general provider binding or create a new general provider binding to meet your business needs. Likewise, if you do not want to use the provided Client sample as the default service client binding, you can select an existing general client binding or create a new general client binding. To specify your global security (cell) default bindings, use the administrative console and click **Services** > **Policy sets** > **Default**

policy set bindings. For environments with multiple security domains, you can optionally choose the general provider and general client bindings that you want to use as the default bindings for a domain. For more information about default bindings see the topic Setting default policy set bindings.

Procedure

1. To manage system policy set attachments for trust service operations, click **Services** > **Trust service** > **Trust service attachments**. The list displays all endpoints that have at least one operation with a policy set attached as well as Trust Service Defaults. The list also displays the system policy set and the binding for each operation.
2. Select one or more of the following actions to configure the trust service attachments:

   New Attachment

   Opens a new panel where you can specify the service endpoint URL. For each new service endpoint that is specified, four trust service operations (cancel, renew, validate and issue) change from having inherited attachments to being explicitly attached. The four operations are attached to the respective policy set and binding as specified in Trust Service Defaults. These initial attachments can be changed.

   Attach

   Displays a list of existing system policy sets, including the default trust-related system policy sets, to which each of the four trust service operations for a service endpoint can be attached. First, select the operation (for example, Cancel token) and then click **Attach** to display the list of available system policy sets. Select a default or custom system policy set to attach. When you change the policy set attachment, the binding automatically changes to **Default**. Select the operation and click **Assign Binding** to change the binding.

   The pre-configured system policy sets that you can select include:

   - **TrustServiceSecurityDefault**

     This trust policy set specifies the asymmetric algorithm as well as the public and private keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using RSA. Message confidentiality is provided by encrypting the body and signature using RSA. This policy set follows the WS-Security specification for the issue and renew trust operation requests.
   - **TrustServiceSymmetricDefault**

     This trust policy set specifies the symmetric algorithm as well as the derived key algorithms to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using HMAC-SHA1. Message confidentiality is provided by encrypting the body and signature using AES. This policy set follows the WS-Security and WS-SecureConversation specifications for the validate and cancel trust operation requests.
   - **SystemWSSecurityDefault**

     This system policy set specifies the asymmetric algorithm and both the public and private keys to provide message security. Message integrity is provided by digitally signing the body, time stamp, and WS-Addressing headers using RSA encryption. Message confidentiality is provided by encrypting the body and signature using RSA encryption.

   Inherit Operation Defaults

   Sets the operation to inherit the respective trust service default trust service policy set attachment and binding. If you select the attachments to modify and then click **Inherit Operation Defaults**, the explicit attachment for both the policy set and the binding is removed. Thereafter, the operation inherits any change to the default trust service policy set and binding.

   Assign Binding

   Changes the existing binding. You can create and assign a new binding, assign the Default binding, or assign an existing trust service specific binding to each of the selected trust service attachments.

Update Runtime
Updates the trust service runtime with any configuration changes that are made to the trust service attachments, token providers, and targets.

3. Optional: Modify the custom policy set by clicking the name of a custom policy set from the list. Edit the settings for custom policy sets, as needed. Default trust service policy set information can only be viewed.

   You cannot edit the default policy sets: TrustServiceSecurityDefault and TrustServiceSymmetricDefault, or SystemWSSecurityDefault. TrustServiceSecurityDefault is the default for the issue and renew operations. TrustServiceSymmetricDefault is the default for the cancel and validate operations.

   At least one trust service operation for the endpoint service URL must be explicitly attached for the endpoint service URL to be displayed. If an operation is explicitly attached, the system policy set name appears. If no policy set is explicitly attached, the respective default trust service policy set appears, followed by the text `(inherited)`.
4. Optional: Modify the trust service specific binding by clicking the name of a binding from the list, as needed. Edit the settings for the trust service specific binding, as needed. Any modifications to a trust service binding affect all trust service attachments that reference the binding.

   If the resource has a policy set directly attached, either the bindings name appears or `Default` appears.
5. Save your changes before applying the changes to the trust service runtime configuration.
6. Click **Update Runtime** to update the trust service runtime configuration with any data changes for token providers, trust service attachments, and targets. Whether the confirmation window appears depends on whether you select the **Show confirmation for update runtime command** check box. Expand **Preferences** to view the check box.
7. Optional: Confirm or cancel if the confirmation window appears. If you deselected the **Show confirmation for update runtime command** check box, all changes are made immediately without displaying the confirmation window.

Results

You have provided the basic information to create or update a trust service attachment. You have configured trust service operation attachments to system policy sets and bindings.

What to do next

You can also create a new attachment for the WebSphere Application Server trust service using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

Creating a service endpoint attachment using the administrative console:

You can attach the trust service operations for a new service endpoint URL to system policy sets and bindings. The operations for each new endpoint are attached to the Trust Service Default policy sets and bindings. Each new endpoint initially has the following four operations: issue, renew, cancel, and validate.

Before you begin

First you must define your policy sets and their bindings. *Policy sets* describe the protection or quality of service that is provided (such as message security, transport and so forth). *Bindings* specify some details about how to implement the policy set, such as: the path for the keystore file, the class name of the token generator, or the JAAS configuration name.

Important: Only use system policy sets with the trust service. The requestor (client) must utilize only Java API for XML-Based Web Services (JAX-WS). Requestors that use Java API for XML-based remote procedure calls (JAX-RPC) are incompatible with the policy set QOS.

About this task

Attaching the trust service operations for a new endpoint to existing policy sets and bindings requires two steps. After initially attaching the endpoint, the following four operations are configured: issue, renew, cancel, and validate. These four operations explicitly attach to Trust Service Defaults. You can then modify these attachments to existing policy sets and bindings.

This task describes how to create or manage service endpoint URLs that you want to attach to the policy set and binding. To complete the configuration for the WebSphere Application Server trust service, you must also create or manage targets.

If no explicit bindings are attached, WebSphere Application Server uses the cell-level default binding, referred to as Default.

Procedure

1. To view existing trust service attachments, click **Services** > **Trust service** > **Trust service attachments**. Until you create the first attachment, only the default attachments for each operation are displayed.
2. To create an attachment, click **New Attachment**.
3. Enter the service endpoint URL in a valid format. Note that when the URL in the trust service attachment does not match the URL, including matching the case, to which the trust service request is sent, the policy set that is defined in the attachment is not applied. Instead, IBM WebSphere Application Server uses the policy set that is attached to the default for the trust operation.

   For example, where `demo` is the endpoint, you might enter: http://localhost:9080/wssamplebeta/demo
4. Click **Attach** to attach the URL and to return to the Trust service attachments panel. After you click **Attach**, the Trust service attachments panel displays the new service endpoint URL and the initial four operations. The service endpoint URL that you specified is listed in the Trust service attachments collection. These four token operations (cancel, renew, validate and issue) for the specified endpoint are initially attached to Trust Service Defaults.
5. On the Trust service attachments panel, change the policy set or binding attachment, as needed. You can return any operation to its initial state by inheriting Trust Service Defaults.

   Note: Changing the policy set forces the binding to change to Default.
6. Save your changes before applying the changes to the Web Services Security runtime configuration.
7. Click **Update Runtime** to update the Web Services Security runtime configuration with any data changes for token providers, trust service attachments, and targets. Whether the confirmation window appears depends on whether you selected the **Show confirmation for update runtime command** check box. Expand **Preferences** to view the check box.
8. Optional: Confirm or cancel if the confirmation window appears. If you deselected the **Show confirmation for update runtime command** check box, all changes are made immediately without displaying the confirmation window.

Results

You have provided the basic information to create a trust service attachment and to configure a policy set, a binding, and the operation information.

What to do next

You can also create a new attachment for the trust service using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

Next, configure the security context token provider or configure targets to complete the trust service configuration.

Trust service attachments collection:

Use this page to view information about or manage system policy set attachments and bindings. Endpoints with at least one operation directly attached to a policy set are displayed.

This page displays each endpoint that has at least one operation that is directly attached to a system policy set. The operations for other endpoints inherit the trust service default policy set and binding data. You can click **New Attachment** to create explicit attachments for endpoints not displayed, or click **Attach** to change the policy set for an operation. Changing the system policy set for an operation removes the binding data for that operation, and resets that data to the system default binding settings. You can also click **Assign Binding** to create a new binding configuration or change the existing binding configuration for the selected operation.

To view this administrative console page, click **Services > Trust service > Trust service attachments**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Show confirmation for update runtime command:

Specifies whether to enable or disable the display of the confirmation window before the Web Services Security runtime configuration is updated for supported tokens, targets, and trust service attachments.

Click **Preferences** to expand the information. You can select or clear the **Show confirmation for update runtime command** check box. If you do not select this check box, updates to the security runtime configuration are made without first displaying a confirmation window. If you select the check box, the confirmation window is displayed before updates to the security runtime configuration are made.

Data type: Check box
Default: Enabled (check box is selected)

Retain filter criteria:

Specifies whether to retain the filter criteria.

Click **Preferences** to expand the information. You can select or deselect the **Retain filter criteria** check box. This check box determines whether **Endpoint URL** is used as the filter criteria to reduce the displayed list of endpoints.

Data type: String
Default: All (check box is not selected)

Search terms:

Specifies the search criteria to use to reduce the displayed list of endpoints.

Click **Preferences** to expand the information. Type the search term you want to use in the **Search terms** field. Use the asterisk (*) as a wildcard character for all terms. You can also search for multiple unknown or partial characters within the term. For example, typing the search term `par*` returns `partly`, `participate`, `partial`, and all other terms beginning with the letters `par`.

Data type: String
Default: * (search for all)

Select:

Specifies that you want to select an existing resource, such as an endpoint or an operation, for further actions.

For existing endpoints, select the check box next to an operation, and then select one of the following actions:

Actions	Description
Attach	Displays a list of policy sets that are available to be attached to an endpoint operation (cancel, reset, validate, or issue) or to one of the trust service default operations. Highlight and click the policy set to attach the policy set to the selected operation. You cannot attach a policy set to an endpoint.
Inherit operation defaults	Detaches the currently attached policy set and binding for each selected operation and sets the operation to inherit the trust service default policy set and binding for each operation.
Assign binding	Lists the bindings that are available to select for the policy set to which you want to attach the binding. You can also create a new binding. • Select **Default** to create and assign the system default binding to the selected policy set attachment. When you select this binding the runtime uses the default binding for the server, cell or in the multiple security domain environment to which the service resource is deployed. • Select **New Trust Service Specific Binding** to create a binding that is specific to the policy set and shares the characteristics of the policy set. This type of binding is reusable only for trust service attachments. • Select an existing general binding to assign the binding to the selected policy set attachment. Multiple selection is valid only when all the resources have the same policy set attached.

New attachment:

Specifies that you want to create an explicit policy set attachment.

Click **New Attachment** to access a new panel where you can enter an endpoint URL to create attachments for each of the four endpoint operations of the provided URL. Initially, the attachment consists of the policy set and binding that are listed as the Trust Service Default for that operation.

Data type: Button

Update runtime:

Updates the trust service configuration for any changed attachments, targets, and token information.

If the **Show confirmation for update runtime command** preference is enabled, then a panel is displayed where you can confirm that you want to update the trust service configuration. If the preference is disabled, the trust service configuration is updated immediately without any confirmation.

Data type: Button

Service endpoint URL / Operation:

Displays a list of the trust service default operation attachments and every service endpoint URL that has at least one operation with a policy set attached.

Each endpoint has four operations: issue, cancel, renew, and validate. Each of the operations for all other endpoints inherits the trust service default policy set and binding.

When the URL in the trust service attachment does not match the URL to which the trust service request is sent, the policy set that is defined in the attachment is not applied. Instead, IBM WebSphere Application Server uses the policy set that is attached to the default for the trust operation.

Data type: String
Default: Trust Service Default

Policy set:

Displays the attached or inherited policy set for each operation of all endpoint URLs. Any endpoint URL that is not displayed inherits the trust service default policy set for each operation. Provides a list of default and custom system policy sets that are attached to the service endpoint URL.

The policy set names are displayed in this column for each operation. If the policy set is inherited from the trust service default, rather than being explicitly attached, **inherited** is displayed in parentheses following the policy set name. Because only operations can have a policy set attachment, the Policy Set column for each endpoint URL row displays **Not applicable**.

Click the system policy set name to view or edit the policy set details information. Note that you can view, but not edit, the default policy sets. Default policy sets cannot be changed.

Data type: String
Defaults: TrustServiceSecurityDefault, TrustServiceSymmetricDefault or SystemWSSecurityDefault

Binding:

Displays the binding that is assigned to each policy set attachment for each operation of the listed endpoint URLs. Any endpoint URL that is not displayed inherits the trust service default binding for each of the four operations.

The name of the assigned binding for each policy set attachment is displayed in this column for each operation. If the attachment is inherited from the trust service default, **inherited** is displayed in parentheses following the binding name. If you select **Assign Binding > Default**, the system default binding is applied to the policy set attachment, and the word **Default** is displayed in this column. If the system default binding is inherited, then **inherited** is displayed in parentheses following **Default**.

The system default binding is also assigned when you attach a new policy set to an operation. Because only operations can have policy set attachments, the binding column for each endpoint URL row displays **Not applicable**. Rows that are not directly related to a token and display the trust service default, display the text, **Not applicable**, for the binding. Additionally, rows that are not directly related to a token and display only the service endpoint URL display the text, **Not applicable**, for the binding.

Click the trust service specific binding name to view or edit the binding information. You can view, but not edit, the TrustServiceSecurityDefault, TrustServiceSymmetricDefault or SystemWSSecurityDefault bindings.

Data type: String
Default: TrustServiceSecurityDefault, TrustServiceSymmetricDefault or SystemWSSecurityDefault

Trust service attachments settings:

Use this page to create a new attachment to the current Trust Service Defaults policy set and binding for the four token operations: cancel, issue, renew, and validate.

To view this administrative console page, complete the following procedure:

- Click **Services** > **Trust service** > **Trust service attachments** > **New Attachment** .

Service endpoint URL:

Specifies the service endpoint URL that you want to attach to the policy set and binding for the trust service default operations.

Use this field to specify a service endpoint URL. The URL must be specified in a valid format, such as `http://www.mybusiness.com`.

Note that when the URL in the trust service attachment does not match the URL to which the trust service request is sent, the policy set that is defined in the attachment is not applied. Instead, IBM WebSphere Application Server uses the policy set that is attached to the default for the trust operation.

After you enter the URL and click **Attach**, the custom service endpoint URL is displayed in a list of explicitly attached service endpoint URLs on the Trust service attachments panel. In addition to the new service endpoint URL, the Trust service attachments panel displays a list of the corresponding four operations (cancel, issue, renew and validate).

On the Trust service attachments panel, you can change the Trust Service Default policy set and binding attachments for any of the four operations. These policy sets apply to any URL not displayed, and therefore not explicitly attached to a policy set and binding. Changing the policy set for a URL operation resets a custom binding setting to the default value.

On the Trust service attachments panel, if you want to remove the explicit policy set attachments and binding assignments, select each of the URL operations, and click **Inherit Operation Defaults**. If all four operations are changed to inherit the Trust Service Default policy set and binding, then the URL no longer displays on this panel.

Data type: String (URL format)

Configuring the security context token provider for the trust service using the administrative console:

Configure the WebSphere Application Server trust service to issue a specific security token to the requestor for communication with an endpoint. Use the administrative console to configure the security context token provider that the trust service provides.

Before you begin

WebSphere Application Server provides a trust service. The trust service provides both a security token service and additional WebSphere Application Server trust-related functionality. To configure the trust service, in addition to managing the security context token provider, you must first complete the following tasks:

- Create or manage supported targets. You can create explicit assignments for new service endpoints (targets) or manage endpoints that have the security context token provider explicitly assigned or that inherit the token provider designated as the Trust Service default.
- Create or manage the attachment of token operations for service endpoints to policy sets and bindings.

The order in which you complete these tasks is not important.

About this task

This task describes how to manage the security context token provider and how to define or modify the properties of the security context token provider.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. To manage the security context token provider, click **Services > Trust service > Token providers**.
2. To edit the settings of the security context token provider configuration, click the link for the token provider name. You cannot edit the name, class name, or token type schema URI when modifying the token provider information.
   a. The format of the token type schema Uniform Resource Identifier (URI) is in the standard URI format. For example, for a version 1.3 security context token, the URI is: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct
   b. Change the amount of time, in minutes, in the **Time in cache after timeout** field that the expired token is kept in cache and where the token can still be renewed. The default value is 120 minutes. This value cannot be less than 10 minutes.
   c. Change the amount of time, in minutes, in the **Token timeout** field that the issued token is valid. The default value is 10 minutes. This value cannot be less than 10 minutes.
   d. Select the **Allow renewal after timeout** check box to enable the renewal of a token after the token has expired. If selected, the amount of time, within which an expired token can still be renewed, is specified in minutes in the **Time in cache after expiration** field.
   e. Select the **Allow postdated tokens** check box to enable postdated tokens. Use postdated tokens to specify whether a client can request a token to become valid at a later time.
   f. Select the **Support Secure Conversation Token v200502** to enable use of the older draft submission specification level of the security context token. The correct URI for this level of the token type schema appears in the field under the check box: http://schemas.xmlsoap.org/ws/2005/02/sc/sct.
   g. Click **New** to define a new custom property or click **Edit** to modify the custom property. Specify these settings using the Custom Properties setting. Custom properties are used to set internal system configuration properties. Custom properties are arbitrary name-value pairs of data, where the name might be a property key or a class implementation, and where the value might be a string or the value might be a true or false value.
   h. If you define a custom property, type a name. Refer to the documentation for the token provider for valid custom property names.
   i. If you define a custom property, type a value. Refer to the documentation for the token provider for the values for a property name.

j. Repeat defining the name and the value for each custom property that you add.
k. Click **OK**. You are returned to the Token providers panel.

3. Save your changes before applying the changes to the Web Services Security runtime configuration.
4. Click **Update Runtime** to update the Web Services Security runtime configuration with any data changes for token providers, trust service attachments, and targets. Whether the confirmation window is displayed depends on whether you select the **Show confirmation for update runtime command** check box. Expand **Preferences** to view the check box.
5. Optional: Confirm or click **Cancel** when the confirmation window appears. If you deselected the **Show confirmation for update runtime command** check box, all changes are made immediately without displaying the confirmation window.

Results

You have completed the required steps to modify the security context token provider configuration and to update the Web Services Security runtime configuration. You can also update the security context token provider configuration for the trust service using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

What to do next

Next, if you have not done so already, you must also configure targets or configure attachments to complete the trust service configuration.

Modifying the security context token provider configuration for the trust service using the administrative console:

WebSphere Application Server provides a pre-configured token, the Security Context Token (SCT). Use the administrative console to modify the configuration of the security context token provider.

Before you begin

WebSphere Application Server provides a trust service. The trust service provides both a security token service and additional WebSphere Application Server trust-related functionality. To configure the trust service, in addition to managing the security context token provider, you must first complete the following tasks:

- Create or manage supported targets. You can create explicit assignments for new service endpoints (targets) or manage endpoints that have a security token provider explicitly assigned or that inherit the token provider designated as the Trust Service default.
- Create or manage the attachment of token operations for service endpoints to policy sets and bindings.

The order in which you complete these tasks is not important.

About this task

This task describes how to configure the security context token provider and how to define the token provider properties.

Procedure

1. To configure the security context token provider, click **Services > Trust services > Token providers**.
2. To change the configuration of the security context token provider, click the link for the token provider name (Security Context Token). For an existing token, the token name, class name and URI are displayed, but are not editable.

3. Optional: Change the amount of time, in minutes, in the **Time in cache after expiration** field that the expired token is kept in cache and where the token can still be renewed. The default value is 120 minutes, and you cannot type a value that is less than 10 minutes.
4. Optional: Change the amount of time, in minutes, in the **Token timeout** field that the issued token is valid. The default value is 120 minutes, and you cannot type a value that is less than 10 minutes.
5. Optional: Select the **Allow renewal after timeout** check box to enable the renewal of a token, after the timeout time has expired. If selected, the amount of time, within which an expired token can still be renewed, is specified in the **Time in cache after expiration** field.
6. Optional: Select the **Allow postdated tokens** check box to enable postdated tokens. Use postdated tokens to specify whether a client can request a token to become valid at a later time.
7. Optional: Select the **Support Secure Conversation Token v200502** check box to enable use of the older draft submission specification level of the security context token. The correct URI for this level of the token type schema appears in the field under the check box: http://schemas.xmlsoap.org/ws/2005/02/sc/sct.
8. Click **New** if you want to define a new custom property. Specify additional configuration using the **Custom Properties** setting. Custom properties are used to set internal system configuration properties. Custom properties are arbitrary name-value pairs of data, where the name might be a property key or a class implementation, and where the value might be a string or Boolean value.
   a. If defining a new custom property, type a name. For example, for a custom property, type: com.ibm.wsspi.wssecurity.trust.keySize
   b. If defining a new custom property, type a value. For example, the following value: 128
   c. Repeat the name and value steps for each new custom property.
9. Click **OK**. You are returned to the Token provider panel.
10. Save your changes before applying the changes to the Web Services Security runtime configuration.
11. On the Token provider panel, click **Update Runtime** to update the Web Services Security runtime configuration with any data changes for token providers, trust service attachments, and targets. Whether the confirmation window is displayed depends on whether you select the **Show confirmation for update runtime command** check box. Expand **Preferences** to view the check box.
12. Optional: Confirm or click **Cancel** when the confirmation window appears. If you deselected the **Show confirmation for update runtime command** check box, all changes are made immediately without displaying the confirmation window.

Results

You have completed the required steps to modify the configuration of the security context token provider and to update the Web Services Security runtime configuration. You can also modify the configuration of the security context token provider for the trust service using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

What to do next

If you have not done so already, you must also configure targets or configure attachments to complete the trust service configuration.

Trust service token custom properties:

WebSphere Application Server trust service provides several custom properties by default to define the default security context token (SCT).

Custom properties are name-value pairs of data that are passed to the token provider during configuration.

The Property name column displays the name of the custom property. The name must match the name of a configuration property or setting that the provider understands and expects. The Property value column displays the configuration setting that is passed to the provider during configuration.

These custom properties are provided by default by WebSphere Application Server, for you to configure when using the **Services > Trust service > Token providers > Security Context Token** page.

algorithm:

The value is AES.

keySize:

The value is 128.

Provider:

The value is IBMJCE.

Disabling the submission draft level for the security context token provider:

Use the administrative console to configure the security context token provider that the trust service provides. Two levels of the token are supported on WebSphere Application Server: the token defined by the WS-Trust February 2005 Submission Draft specification, and the token defined by the OASIS WS-Trust Standard version 1.3. You can disable a setting so that the server will not accept a trust request that specifies the submission draft level of the token.

About this task

Disable the Security Context Token provider support for the submission draft specification using the administrative console.

Procedure

1. Log on to the administrative console and navigate to the Token providers panel by clicking **Services** > **Trust service** > **Token providers**.
2. Click on **Security Context Token**.
3. Click to clear the **Support Secure Conversation Token v200502** check box.
4. Click **Apply** to save the changed setting.

Results

For more information, see the topic Modifying the security context token provider configuration for the trust service using the administrative console.

Trust service token provider settings:

Use this page to modify information for an existing token provider.

To view this administrative console page, complete the following actions:

- **Services** > **Trust service** > **Token providers** > ***token_provider_name***

Name:

Specifies the name of the token provider.

This field displays the unique name of the token provider (for example, `Security Context Token`). You cannot change the name for any existing token provider.

Data type: String

Class name:

Specifies the package and class name of the trust service's Security Context Token provider.

This field displays the configuration class name, including the package information (for example, `com.ibm.ws.wssecurity.trust.server.sts.ext.sct.SCTHandlerFactory`).

You cannot change the class name for any existing token provider.

Data type: String

Token type schema URI:

Specifies the Uniform Resource Identifier (URI) for the token type schema.

This field displays the unique token type schema URI. Use a valid URI format, such as: `http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct`.

You cannot change the schema URI for any existing token provider.

Data type: String

Time in cache after expiration:

Specifies the number of minutes that a token remains in the token cache after the token expires.

This field displays the time, in minutes, that the expired token is kept cached and can still be renewed.

Data type: Integer
Default: 120
Minimum: 10
Maximum: 2147483647

Token timeout:

Specifies the amount of time, in minutes, that the issued token is valid.

This field displays the maximum timeout, in minutes, for a token to be considered valid.

Data type: Integer
Default: 120
Minimum: 10
Maximum: 2147483647

Allow renewal after timeout:

Specifies to enable or disable the renewal of a token.

This check box specifies whether to allow a client to renew an expired token. Note the **Time in cache after expiration** field specifies the amount of time within which an expired token can still be renewed.

Data type:	Check box
Default:	Do not allow (unchecked)

Allow postdated tokens:

Specifies to enable or disable the use of postdated tokens.

This check box specifies whether a client can request a token to become valid at some point in the future.

Data type:	Check box
Default:	Do not allow (unchecked)

Support Secure Conversation Token v200502: This check box specifies whether support for the WS-Trust and WS-Secure Conversation Feb 2005 Submission Draft OASIS specification is enabled. The default URI for the token type schema is provided in the non-editable field below the check box.

Data type:	Check box
Default:	Enabled (checked)

Custom Properties:

Specifies additional configuration settings that the token provider might require.

This table lists custom properties. Use custom properties to set internal system configuration properties.

The Secure Context Token default configuration settings are :

Property Name	Property Value
com.ibm.wsspi.wssecurity.trust.algorithm	AES
com.ibm.wsspi.wssecurity.trust.keySize	128
com.ibm.wsspi.wssecurity.trust.provider	IBMJCE

Select:

Specifies custom properties that you can add to, edit, or delete from the token provider.

Click **New** to add and define a new custom property.

For existing custom properties, first select the check box for the name of the custom property, and click one of the following actions:

Actions	Description
Edit	Specifies whether to modify existing custom properties. This action requires one or more custom properties to be selected.
Delete	Removes the selected existing property from the listing in the Name column. This action requires one or more custom properties to be selected.

Name:

Displays the names of the custom properties that have been defined for the token provider.

This column displays the name of the custom property (for example, `com.ibm.wsspi.wssecurity.trust.keySize`). Custom properties are name-value pairs of data that are passed to the token provider during configuration. The name that you specify must match the name of a configuration property or setting that the provider understands and expects.

Data type: String

Value:

Specifies the value for the custom property.

This column displays the value for the custom property (for example, `true`). Custom properties are name-value pairs of data. The value, which is represented as a string, is a configuration setting that is passed to the provider during configuration.

Data type: String or Boolean

Trust service token providers collection:

Use this page to view information about or manage token providers for the trust service.

To view this administrative console page, click **Services > Trust service > Token providers**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Show confirmation for update runtime command:

Specifies to enable or disable the display of the confirmation window before the trust service configuration is updated when you click **Update Runtime**.

Click **Preferences** and then select the **Show confirmation for update runtime command** check box. If you select this check box, the confirmation window is displayed before updates to the security trust service configuration are made. If you do not select this check box, clicking **Update Runtime** updates the security trust service configuration without first displaying a confirmation window.

Data type: Check box
Default: Enabled (checked)

Update Runtime:

Updates the trust service configuration for any changed attachments, targets, and token information.

If the **Show confirmation for update runtime command** preference is enabled, then a panel is displayed where you can confirm that you want to update the trust service configuration. If the preference is disabled, updates to the trust service configuration are applied immediately without any confirmation.

Data type: Button

Token Provider Name:

Lists available token providers.

This column displays the names of the pre-configured token providers. The pre-configured token provider is the Security Context Token (SCT). Click a token provider name link to view additional details.

Data type: String
Default: Security Context Token

Token Type Schema URI:

Provides the Uniform Resource Identifier (URI) for the token type schema.

This column displays the URIs of all pre-configured token providers.

Data type: String

Configuring trust service endpoint targets using the administrative console:

The Trust Service manages tokens on behalf of service endpoints. A token provider is either explicitly or implicitly associated with each service endpoint. A specific token can be explicitly assigned to be issued when access to an endpoint is requested. Otherwise, the Trust Service Default token is issued.

Before you begin

The Web Services Secure Conversation specification defines the protocol for a client to establish a secure session with a target service. The security token service that WebSphere Application Server provides, referred to as the trust service, issues only the Security Context Token (SCT). The security context token is used for Web Services Secure Conversation (WS-SecureConversation).

About this task

This task describes how to create new or manage existing assignments of tokens to be issued for endpoint targets. You can create explicit assignments for new service endpoints (targets) or manage existing token assignments.

To complete the configuration for the trust service, you must have performed the following tasks:

- Manage the security context token provider.
- Create or manage service endpoint URLs that you want to attach to the policy set and binding.

The order in which you complete these tasks is not important.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Procedure

1. To configure new and existing trust service endpoint targets, click **Services > Trust service > Targets**. A list of all service endpoints that have a security token provider explicitly defined is displayed. The token provider assigned to the Trust Service Default by default handles requests to issue tokens to access an endpoint.
2. Click one of the following actions to manage a new or existing endpoint target configuration:

New Assignment

Opens a new panel where you can specify a custom service endpoint URL and explicitly assign the token provider, which is specified as the Trust Service Default, to be issued for access to the endpoint.

Change Token

Changes an explicitly assigned token to be issued for the service endpoint to the security context token. Select an endpoint and then click **Change Token**. Select the Security Context Token.

Also, removes the explicit assignment of a token to be issued; therefore, the token that is issued is inherited from the Trust Service Default. Select an endpoint and then click **Change Token**. Click **Inherit Default** to remove a token provider assignment for the selected endpoint and to return the issued token to be the token that is specified as the Trust Service Default. If the token that is issued is inherited, the endpoint is no longer displayed in the list because the token provider is no longer explicitly assigned to the endpoint.

3. Click the token name link for an existing endpoint target to modify the token provider configuration information. You can modify the token type schema URI, or change custom properties.
4. Save your changes before applying the changes to the Web Services Security runtime configuration.
5. Click **Update Runtime** to update the Web Services Security runtime configuration with any data changes for token providers, trust service attachments, and targets. Whether the confirmation window is displayed depends on whether you select the **Show confirmation for update runtime command** check box. Expand **Preferences** to view the check box.
6. Optional: Confirm or click **Cancel** when the confirmation window appears. If you deselected the **Show confirmation for update runtime command** check box, all changes are made immediately without displaying the confirmation window.

Results

When you complete these steps, the service endpoint URL displays in the Targets collection, unless you changed the token to inherit the default value. You can also configure the trust service to issue tokens for individual endpoint targets using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

What to do next

You have completed the required steps to create or manage existing trust service targets, to assign the security token provider to an endpoint target, and to update the Web Services Security runtime configuration. Next, if you have not competed these tasks already, configure the security context token provider or configure attachments to the policy set and binding to complete the trust service configuration.

Assigning a new target for the trust service using the administrative console:

You can associate a security token provider with a service endpoint using the administrative console. After entering the service endpoint URL, the token provider configured as the Trust Service Default is explicitly associated with the service endpoint.

Before you begin

The Web Services Secure Conversation specification defines the protocol for a client to establish a secure session with a target service. The security token service that WebSphere Application Server provides, referred to as the trust service, issues the Security Context Token (SCT). The security context token is required for Web Services Secure Conversation (WS-SecureConversation).

About this task

This task describes how to register a service endpoint (target) with the trust service. Registration of an service endpoint with the trust service initially associates the token provider configured as the Trust Service Default with that service endpoint.

To complete the configuration for the trust service, you must have completed the following tasks:

- Manage the Security Context Token.
- Create or manage service endpoint URLs that you want to attach to the policy set and binding.

The order in which you complete these tasks is not important.

Procedure

1. To configure a custom endpoint target, click **Services > Trust service > Targets > New Assignment**.
2. At the New assignment panel, enter the Universal Resource Locator (URL) for the service endpoint, and click **Assign**. You are returned to the Targets panel where the custom service endpoint URL is displayed in the list. Initially, the token that is explicitly assigned to the custom endpoint is the token that is assigned as the Trust Service Default.
3. At the Targets panel, select the check box for a service endpoint, click **Change Token**, and select one of the following:
   a. Security Context Token (SCT). A security context token is defined by the WS-SecureConversation specification.
   b. **Inherit Default** if you want the token that is issued to be the token assigned as the Trust Service Default. The endpoint is not displayed in the list when the assignment is inherited because the token is no longer explicitly assigned to the endpoint.
4. At the targets panel, click the token name link for an existing endpoint target to modify the token provider configuration information.
5. Save your changes before applying the changes to the Web Services Security runtime configuration.
6. Click **Update Runtime** to update the Web Services Security runtime configuration with any data changes for token providers, trust service attachments, and targets. Whether the confirmation window is displayed depends on whether you select the **Show confirmation for update runtime command** check box. Expand **Preferences** to view the check box.
7. Optional: Confirm or click **Cancel** when the confirmation window appears. If you deselected the **Show confirmation for update runtime command** check box, all changes are made immediately without displaying the confirmation window.

Results

When you complete these steps, service endpoints explicitly associated with a token provider are displayed in the Targets collection. Service endpoints that have been changed to inherit the token provider configured as the Trust Service Default are not displayed. You can also configure the security token service to issue a specific token for access to a target using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

What to do next

You have completed the required steps to create a service endpoint URL, to assign the token to be issued for access to the target, and to update the Web Services Security runtime configuration. Next, if you have not completed these tasks already, configure the Security Context Token provider or configure attachments to the policy set and binding to complete the trust service configuration.

Trust service targets collection:

Use this page to view a list of targets, which are application server service endpoints. You can manage tokens by specifying which token is to be issued when access to a specific endpoint is requested.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

To view this administrative console page, click **Services** > **Trust service** > **Targets**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Show confirmation for update runtime command:

Specifies to enable or disable the display of the confirmation window before the WebSphere Application Server trust service configuration is updated when you click **Update Runtime**.

Click **Preferences** and then select the **Show confirmation for update runtime command** check box. If you select this check box, the confirmation window is displayed before updates to the trust service configuration are made. If you do not select this check box, clicking Update Runtime updates the trust service configuration without first displaying a confirmation window.

Data type: Check box
Default: Enabled (checked)

Select:

Specifies a check box for the service endpoint Universal Resource Locator (URL) that you want to select for further actions.

For existing endpoints, select the checkbox for the service endpoint and select one of the following actions:

Actions	Description
Change Token	Changes the token that is issued when access to an endpoint is requested. Selecting **Inherit Default** in the Change Token menu causes the following actions to occur: • The security token assignment is removed for the endpoint. • The token assigned as the Trust Service Default is issued for access to the endpoint. • The endpoint is no longer displayed in the list of endpoints that have tokens explicitly assigned. Only endpoints that are explicitly assigned a security token are displayed in the list. Endpoints that inherit the default do not display in the list.

New Assignment:

Defines a new service endpoint.

Initially, each endpoint is explicitly assigned the Trust Service Default token. By default, the pre-configured Security Context Token (SCT) is assigned, but that can be changed.

Data type: Button

Update Runtime:

Updates the trust service configuration for any changed attachments, targets, and token information.

If the **Show confirmation for update runtime command** preference is enabled, then a panel is displayed where you can confirm that you want to update the trust service configuration. If the preference is disabled, updates the trust service configuration immediately without any confirmation.

Data type: Button

Service Endpoint URL:

Specifies the Universal Resource Locator (URL) of the service endpoint for the explicitly assigned token.

This column lists the default service endpoint, Trust Service Default, and any custom service endpoints that have a token that is explicitly assigned to the endpoint, such as: `http://localhost:9080/EcommerceSTS`.

Data type: String
Default: Trust Service Default

Token Name:

Displays the name of the token to be issued when access to the endpoint is requested.

To inherit the default token, select the check box for a custom service endpoint URL, click **Change Token** > **Inherit Default**.

You can change the token type that is explicitly assigned as the Trust Service Default, but the token type cannot be left unassigned. If the token is not explicitly assigned, then the endpoint inherits the token that is assigned as the Trust Service Default token.

Click a token name link to access detailed information about the token. You can modify the token information, except for the token name. It is recommended that you do not modify the class name or the token type schema URI for the default token type, Security Context Token.

Changes to token properties apply to all tokens of this type that are issued for any endpoint.

Data type: String
Default: Security Context Token

Token Type Schema URI:

Specifies the schema Uniform Resource Identifier (URI) for the token type.

This column displays the schema URI for the explicitly assigned token type (for example, Security Context Token) in a valid URI format. The token type schema URI is a property of the token name and describes the version of the specification that is implemented for the security token.

Data type: String
Default value: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/sct

Trust service targets settings:

Use this page to specify a custom service endpoint Universal Resource Locator (URL) and to assign a custom token type to the endpoint URL.

To view this administrative console page, click **Services** > **Trust service** > **Targets** > **New Assignment**.

Service endpoint URL:

Specifies the URL for the service endpoint.

Use this field to specify a custom service endpoint URL. The URL must be specified in a valid format, such as `http://localhost:9080/EcommerceSTS`. After you enter the URL and click **Assign**, the endpoint URL is explicitly assigned to the security token that is assigned the Trust Service Default.

The service endpoint URL is added to the list that displays on the Targets panel. Only endpoints that are explicitly assigned a security token are displayed in the list. Endpoints that inherit the default do not display in the list.

By default, the Trust Service Default token is the Security Context Token (SCT).

After clicking **Assign** and returning to the Targets panel, if you want to remove the explicit token assignment (and thereby change the token to be issued back to the default value), select the custom endpoint URL, and click **Inherit Default**. Then the following actions occur:

- The security token assignment is removed for the endpoint.
- The token assigned as the Trust Service Default is issued for access to the endpoint.
- The endpoint is no longer displayed in the list of endpoints that have tokens explicitly assigned.

Data type: String (URL format)

Updating the Web Services Security runtime configuration:

Update Web Services Security runtime configuration with any data changes that you make and save for token providers, trust service attachments, and targets.

Before you begin

Before you update the Web Services Security runtime configuration, make your required data changes for token providers, trust service attachments, and targets. Save your changes before applying the changes to the Web Services Security runtime configuration.

About this task

Whether the confirmation window appears depends on whether the **Show confirmation for update runtime command** check box is selected. Expand **Preferences** from the Token providers panel, the Trust service attachments panel, or the Targets panel to see the **Show confirmation for update runtime command** check box. Preferences are collapsed by default.

- If you select the check box, then the confirmation window is displayed before updates to the security runtime configuration are made. The check box is selected by default.

- If you do not select the check box, then updates to the security runtime configuration are made immediately, without first displaying the confirmation window. The confirmation window does not appear again until you re-select the checkbox located under Preferences.

  Or, instead of deselecting the check box under Preferences, you can select the **Do not show this message again** check box on the Update runtime confirmation panel.

Procedure

1. Perform one of the following tasks:
   - **Services > Trust service > Trust service attachments**
   - **Services > Trust service > Token providers**
   - **Services > Trust service > Targets**
2. Click **Update Runtime**. If you did not select the **Show confirmation for update runtime command** check box, all changes are made immediately without displaying the confirmation window.
3. Optional: Confirm or cancel when the confirmation window appears. If you selected the **Show confirmation for update runtime command** check box, all changes require confirmation and the confirmation window is displayed.

Results

After clicking **Update Runtime**, the configuration changes you made on the Token providers, the Trust service attachments, or the Targets panel are updated for the Web Services Security runtime configuration.

What to do next

You can also manage and administer the trust service using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

After configuring the trust service targets, attachments, and tokens, next configure the secure conversation client cache, if you have not already completed this step.

Web services update runtime settings:

Use this page to confirm that the web services trust service runtime should be updated with the most recent configuration changes.

To view this administrative console page, complete one of the following procedures:

- **Services > Trust service > Trust service attachments**
- **Services > Trust service > Supported tokens**
- **Services > Trust service > Targets**

Make changes to attachments, tokens, or targets, save the changes, and click .**Update Runtime**.

Do not show this message again:

Specifies to enable or disable the confirmation panel before the Web Services Security runtime configuration is updated.

Click **OK** to confirm that you want to make the configuration changes effective immediately for the web services trust service runtime.

Data type:	Check box
Default	Enabled (checked)

Configuring the Web Services Security distributed cache using the administrative console:

You can configure the Web Services Security runtime to use the security distributed cache to store security tokens.

About this task

Web Services Security functions such as secure conversation, trust, and nonce use the distributed cache to store security tokens when the distributed cache is enabled. If the distributed cache option is not selected, then a local cache is used to store the tokens. WebSphere Application Server Version supports distributed caching for the tokens in both cluster and non-cluster environments. In a cluster environment, you can configure the security cache to be distributed. If the cache is distributed, then all servers in the cluster share information about issued tokens.

Procedure

1. To configure the secure conversation client cache, click **Services** > **Security cache**.
2. Change the time in minutes in the **Time token is in cache after timeout** field. The default value is 120 minutes. The minimum allowable time is 10 minutes, meaning you cannot enter a value that is less than 10 minutes. This field specifies the number of minutes that the token is in cache after the token expiration time expires (cache persist period).
3. Change the time in minutes in the **Renewal interval before token timeout** field. The default value and minimum allowable time is 10 minutes. You cannot enter a value that is less than 10 minutes. This field specifies the time period before the token expires when the client attempts to renew the token. This window of time is just before token expires where, if the token is accessed, then the client attempts to renew the token so that a downstream call can complete.

   It is important that this setting be set to a length of time that is longer than the longest possible transaction. This value must include the time it takes to transport to and from the server, the time that is needed by the server to process the request, and the time that is cached by reliable messaging, if appropriate. Setting this value to a length of time that is too small might result in the token expiring in the middle of a transaction and might prevent the transaction from completing.

   If the Security Context Token is renewed too often, it might cause Web Services Secure Conversation (WS-SecureConversation) to fail or even cause an out-of-memory error to occur. It is required that you set the renewal interval before the token expires value for the Secure conversation client cache to a value less than the token timeout value for the Security Context Token. It is also suggested that the token timeout value be at least two times the renewal interval before the token expires value.
4. Select the **Enable distributed caching** check box, if you want to share the tokens across the cluster. When the checkbox is selected to enable distributed caching, choose one of the following settings for updating the caches.
   - Synchronous update of cluster members: performs synchronous update of cache objects on cluster members (default).
   - Asynchronous update of cluster members: performs a non-synchronous update of the cache on cluster members. This setting allows interoperability with cluster members that use the older style of updating as implemented in versions of WebSphere Application Server prior to version 7.0.
   - Token recovery support: assigns a shared data source as the distributed cache.

   If token recovery support is selected as the update method, then you must select a cell level data source using the drop-down list. Token state data is saved in the database defined as the data source. If there are no available data sources in the list, click on Manage data sources to add one or more new data source objects. The data source object supplies an application with connections for accessing the database.
5. To create a new custom property, click **New**. For example, you might add the `cancelActionRST` custom property with a value of `http://schemas.xmlsoap.org/ws/2005/02/trust/RST/SCT/Cancel`.
6. To edit an existing custom property, select the check box for the name of the existing custom property, and then click **Edit**. For example, you might change the name or the value of the cancelActionRST custom property.

7. Click **Apply** to save and apply the changes.

Results

You have provided the basic information to configure the Web Services Security distributed cache. Use either the administrative console or the wsadmin tool to modify the security cache configuration.

What to do next

You can also add or delete custom properties for the trust service using the wsadmin tool. The wsadmin tool examples are written in the Jython scripting language.

Security cache settings:

Use this page to configure the Web Services Secure Conversation (WS-SecureConversation) security local and distributed cache settings using the administrative console.

To view this administrative console page, click **Services** > **Security cache**.

Time token is in cache after timeout:

Sets the time that the token remains in cache after the token times out.

This field specifies the number of minutes for the time the token is in cache after the token expiration time expires (cache persist period). For example, if you specify 30 minutes, the token is kept in cache for this time period after the token expiration time. The default value is 10 minutes, which is the minimum number of minutes that is allowed.

Data type: Integer
Default: 10 (minutes)

Renewal interval before token timeout:

Sets the time period before expiration that the client attempts to renew the token.

This field specifies the period of time, in minutes, before expiration that the client attempts to renew the token. This setting must specify a period of time that is longer than the time for the longest transaction or else the token might expire during the transaction. This time must include time for transport to and from the server, processing by the server, and any time delay that is because of time used for reliable messaging, when applicable. The default value is 10 minutes, which is the minimum number of minutes that is allowed.

If the Security Context Token is renewed too often, it might cause Web Services Secure Conversation (WS-SecureConversation) to fail or even cause an out-of-memory error to occur. It is required that you set the renewal interval before the token expires value for the security cache to a value less than the token timeout value for the Security Context Token. It is also suggested that the token timeout value be at least two times the renewal interval before the token expires value.

Data type: Integer
Default: 10 (minutes)

Enable distributed caching:

Specifies whether distributed caching is enabled or disabled. If distributed caching is enabled, select distributed cache settings.

Use this check box to specify whether to use distributed caching when the server is in a clustered environment and when the tokens are shared across the cluster.

Data type:	Check box
Default:	No distributed caching (unchecked)

When the checkbox is selected to enable distributed caching, choose one of the following settings for updating the caches.

Button	Resulting Action
Synchronous update of cluster members	Performs synchronous update of cache objects on cluster members (default).
Asynchronous update of cluster members	Performs a non-synchronous update of the cache on cluster members. This setting allows interoperability with cluster members that use the older style of updating as implemented in versions of IBM WebSphere Application Server prior to version 7.0.
Token recovery support	Assigns a shared data source as the distributed cache.

If token recovery support is selected as the update method, then you must select a cell level data source using the drop-down list. Token state data is saved in the database defined as the data source. If there are no available data sources in the list, click on **Manage data sources** to add one or more new data source objects. The data source object supplies an application with connections for accessing the database.

Custom Properties:

Specifies additional configuration settings that the secure conversation client might require.

This table lists custom properties. Use custom properties to set internal system configuration properties. This collection is empty until the first custom property is defined.

Data type:	String

Select:

Specifies that you want to select further actions.

Use this check box to select custom properties for further actions. To manage existing custom properties, select the check box beside the name, and then select one of the following actions:

Actions	Description
Edit	Select to modify an existing custom property. This action is not displayed until you have added at least one custom property.
Delete	Select to remove an existing custom property.

Data type:	Check box

New:

Specifies that you want to add and define a new custom property.

Click **New** to define a new custom property.

Data type: Button

Name:

Lists available custom properties.

This column displays the names of the custom properties that you can use with the secure conversation client (for example, `exampleProperty`). Custom properties are name-value pairs of data, where the name is a string representation of a property that is expected by the secure conversation client.

Data type: String

Value:

Lists the values of the custom properties.

This column displays the values of the custom properties (for example, `true`). Custom properties are name-value pairs of data, where the value is a string representation of the property setting.

Data type: String

Configuring the Kerberos token for Web Services Security

Use this topic to configure the Kerberos token for message-level Web Services Security.

Before you begin

Before you can use Kerberos with Web Service Security, you must configure Kerberos in the IBM WebSphere Application Server. You do not need to enable Kerberos as the authentication mechanism. However, the Kerberos configuration file, `krb5.conf` or `krb5.ini`, and the Kerberos keytab file, `krb5.keytab`, are required.

The initial setup and configuration processes to use Kerberos with Web Services Security are identical to the configuration processes for using Kerberos with the security function. Therefore, you must set up and configure Kerberos before continuing with the steps in this topic.

The "Kerberos (KRB5) authentication mechanism support for security" topic provides an overview of the Kerberos functionality and provides the initial steps for setting up and configuring Kerberos for authentication purposes. Within this topic, you must complete the steps in the section "Setting up Kerberos as the authentication mechanism for WebSphere Application Server." Use that topic to configure Kerberos, the service principal, and the keytab files. In addition, that topic provides references to the process for configuring Kerberos as the authentication mechanism using the administrative console or commands. You can also find information on how to setup up Kerberos when the Key Distribution Center (KDC) and the Application Server do not use the same user registry.

About this task

The Kerberos token for JAX-WS applications is configured using policy sets and bindings. The JAX-WS application is attached with a custom policy and the Kerberos token is configured as a message protection token or an authentication token.

The implemented Kerberos functionality for Web Services Security also leverages existing tools and frameworks for the Kerberos token profile configuration for authentication and message protection. The support for Kerberos with Web Services Security in the product is based on the OASIS Web Services Security Kerberos Token Profile 1.1 specification.

To configure Kerberos with Web Service Security, complete the following steps:

Procedure

1. Enable the Kerberos token profile for JAX-WS applications.

   The JAX-WS application is attached with a custom policy that has a Kerberos token, which is configured with a message protection token or an authentication token. For more information, see "Configuring the Kerberos token policy set for JAX-WS applications."
2. Select the customized Kerberos token type. You can define key bindings for request message protection and response message protection. You can use the key type, such as the key identifier or security token reference, for the outbound key information. If you use a derived key, use a security token reference in both the outbound and inbound key information. If you use a Kerberos session key, you can use a security token reference in the outbound key information and a key identifier in the inbound key information for the client bindings. Then, use a key identifier in the outbound key information and a security token reference in the inbound key information for the provider bindings.
3. Select the customized Kerberos token types for the token generator or token consumer.
4. Configure the bindings for Kerberos message protection for JAX-WS applications. For more information, see the "Configuring the bindings for message protection for Kerberos" on page 3261.

What to do next

Using this task, you have configured the Kerberos token for WebSphere Application Server.

Configuring the Kerberos token policy set for JAX-WS applications:

Use this topic to enable the Kerberos token policy set for JAX-WS applications.

Before you begin

Prior to beginning this task, you must specify the Kerberos configuration information for IBM WebSphere Application Server. For more information, see Kerberos (KRB5) authentication mechanism support for security.

The configuration model for the Kerberos token enables you to choose from the following existing WebSphere Application Server frameworks:

- For JAX-RPC applications, the deployment descriptor and bindings are used in the configuration. JAX-RPC application includes the deployment descriptor for a Kerberos custom token, which is configured with authentication tokens.
- For JAX-WS applications, the configuration uses a policy set and bindings. The JAX-WS application is attached by a custom policy with the Kerberos token configured with authentication tokens, message protection tokens, or both.

Note: Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

About this task

Complete the following steps to configure the Kerberos token policy set for JAX-WS applications using the administrative console for WebSphere Application Server. In these steps, the Main policy configuation panel references the administrative console panel that is available after you complete the first five steps.

Procedure

1. Expand **Services** > **Policy sets** and click **Application policy sets** > **New** to create a new policy set.
2. Specify a name and a short description for the new policy set and click **Apply**.

3. From the Policies heading, click **Add** and then select the **WS-Security** security policy type.
4. Click **OK** and click **Save** to save the new configuration directly to the master configuration.
5. In the **Policies** field, click **WS-Security** and click **Main policy** on the WS-Security panel to configure the main policy for the Kerberos token policy set.
6. From the Key Symmetry heading, select **Use symmetric tokens** for message protection.
7. Click **Symmetric signature and encryption policies** to configure the Kerberos custom token type or clear the **Message level protection** check box if you are configuring an authentication token only.

   Important: You do not need to configure the request token policy if you are using the Kerberos token for message protection. If you are configuring the authentication token only, proceed to the next step. If you are not configuring the request token policy for the authentication token, skip the next step.
8. On the Main policy configuration panel, configure the policy for the request token if you are configuring the authentication token.
   a. From the Policy Details heading, click **Request token policies**.
   b. Click **Add token type** and select **Custom**.
   c. Specify the name of the custom token in the **Custom token name** field.
   d. Specify the local part value in the **Local part** field. For interoperability with other web services technologies, specify the following local part: `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ`. If you are not concerned with interoperability issues, you can specify one of the following local name values:
      - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ`
      - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510`
      - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510`
      - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120`
      - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120`

      These alternative values depend on the specification level for the Kerberos AP-REQ token that is generated by the Key Distribution Center (KDC). For more information about when to use these values, see Token type settings.
   e. Do not specify a value for the **Namespace URI** field if you are generating a Kerberos token.
   f. Click **OK** and **Save** to save the configuration directly to the master configuration.

   This step completes the configuration process for configuring the request token policy for the authentication token. You do not need to complete the next two steps. Complete the next steps to configure encryption and symmetric signature policies.
9. Return to the main policy configuration panel for the application policy set and click **Symmetric signature and encryption policies** to configure the encryption and symmetric signature policies.
   a. From the Message Integrity heading, click the **Action** menu list beside the **Token type for signing and validating messages** field and select **Custom**.
   b. From the Message Confidentiality heading, select the **Use same token for confidentiality that is used for integrity** option.
   c. Click **OK** and **Save** to save the configuration changes.
   d. From the Message Integrity heading, click the **Action** menu list beside the **Token type for signing and validating messages** field and select **Edit Selected Type Policy**.
   e. Edit the custom token type for the signature and encryption by specifying the local part for the Kerberos custom token.

For example, specify `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ` for the local part value. Do not specify a Namespace URI value.

f. Click **OK** and then click the **Save** link to save the configuration changes.

10. Return to the main policy configuration panel for the application policy set and click **Algorithms for symmetric tokens** to configure the symmetric token algorithm.

a. Select the algorithm suite to use for the symmetric tokens from the **Algorithm suite** menu list. Select the Advanced Encryption Standard (AES) algorithms for a Kerberos token that is compliant with RFC-4120.

The symmetric key wrap, or private key cryptography, algorithms include:

- Triple DES key wrap: http://www.w3.org/2001/04/xmlenc#kw-tripledes
- AES key wrap (aes128): http://www.w3.org/2001/04/xmlenc#kw-aes128
- AES key wrap (aes256): http://www.w3.org/2001/04/xmlenc#kw-aes256

Restriction: To use the 256–bit AES encryption algorithm, you must apply the unlimited jurisdiction policy files. To remain in compliance, see Basic Security Profile compliance tips.

Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

For application server platforms using IBM Developer Kit, Java Technology Edition Version 5, you can obtain unlimited jurisdiction policy files by completing the following steps:

1) Visit the IBM developerWorks: Security Information website.
2) Click **Java 5**.
3) Click **IBM SDK Policy files**.

   The Unrestricted JCE Policy files for SDK 5 website is displayed.
4) Enter your user ID and password or register with IBM to download the policy files. The policy files are downloaded onto your workstation.
5) Re-mount your product HFS as read/only.

For more information on the algorithm suite components, see Algorithms settings.

b. Select either the **Exclusive cannonicalization** or **Inclusive cannonicalization** value for the **Cannonicalization algorithm** menu list. For more information, see XML digital signature.

c. Specify the **XPath 1.0** or **XPathfilter 2.0** version to use from the **XPath version** menu list.

What to do next

Configure the bindings for message protection for Kerberos for JAX-WS applications. For more information, see “Configuring the bindings for message protection for Kerberos.”

Configuring the bindings for message protection for Kerberos:

To set up bindings for message protection with JAX-WS applications, you must create a custom binding. Complete this task to set the bindings for a Kerberos token as defined in the OASIS Web Services Security Specification for Kerberos Token Profile Version 1.1.

Before you begin

You must configure Kerberos for IBM WebSphere Application Server. For more information, see Kerberos (KRB5) authentication mechanism support for security. In addition, you must configure the Kerberos token

policy set for JAX-WS applications. For more information, see Configuring the Kerberos token policy set for JAX-WS applications.

About this task

You can leverage existing frameworks including the policy set and bindings for JAX-WS applications.

You can configure a symmetric protection token or an authentication token. Both symmetric protection token and authentication token configurations use similar configuration data. However, you do not need to configure the authentication token if you intend to use a Kerberos symmetric protection token. For whichever token type you use, configure the token generator and the token consumer as indicated in the following list:

- Symmetric protection token
  - Token generator
  - Token consumer
- Authentication token
  - Token generator
  - Token consumer

Use the administrative console to configure the application-specific bindings to use a Kerberos token in web services message protection.

Procedure

1. Expand **Applications** > **Application Types**.
2. Click **WebSphere enterprise applications** > ***application name*** .
3. From the Web Services Properties heading, click **Service provider policy sets and bindings** to configure the service bindings or click **Service client policy sets and bindings** to configure the client bindings.
4. Select the resource to attach to the Kerberos token policy set and select **Attach Policy Set** > ***policy set name***. To configure the Kerberos token policy set, see “Configuring the Kerberos token policy set for JAX-WS applications” on page 3259.
5. Click **Assign bindings** and select the application-specific binding or select **New Application Specific Binding** to create a new binding. To create a new binding, complete the following actions.
   a. Enter a name for the new binding in the **Binding configuration name** field and optionally enter a description for the binding in the **Description** field.
   b. Click **Add** and select **WS-Security** to specify a new policy set.
   c. Click **Authentication and protection** > **New**.
   d. Optional: Define a symmetric protection token for the token generator.

      Important: If you configure a symmetric protection token for the token generator, you must define a complimentary symmetric protection token for the token consumer.

      1) From the Protection tokens heading, click **New** and select **Token Generator**.
      2) Specify the name of the protection token in the **Name** field.
      3) Select **Custom** from the values in the **Token type** menu list.
      4) Specify the local name value in the **Local name** field.

         For interoperability with other web services technologies, specify the following local name: `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ`. If you are not concerned with interoperability issues, you can specify one of the following local name values:

         - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ`

- http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510
- http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510
- http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120
- http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120

These alternative values depend on the specification level for the Kerberos token that is generated by the Key Distribution Center (KDC). For more information about when to use these values, see Protection token settings (generator or consumer).

5) Do not specify a value for the **Namespace URI** field.
6) Select the **wss.generate.KRB5BST** value from the JAAS login menu list.

   If you have previously defined your own Java Authentication and Authorization Service (JAAS) login module, you can select your login module to handle the Kerberos custom token. To define a custom JAAS login module, click **New Application Login** > **New**, specify an alias for the new module, and click **Apply**. For more information, see Login module settings for Java Authentication and Authorization Service.

   Attention: Although the information in the "Login module settings for Java Authentication and Authorization Service" topic refers to security and not Web Services Security, the configuration for a login module for Web Services Security is identical to security.
7) Specify the token generator custom properties for the target service name, host, and realm.

   The combination of the target service name and host values forms the Service Principal Name (SPN), which represents the target Kerberos service principal name. The Kerberos client requests the initial Kerberos AP_REQ token for the SPN. Specify the following custom properties.

Table 285. Target service custom properties. Use these properties to specify the token generator information.

Name	Value	Type
com.ibm.wsspi.wssecurity.krbtoken.targetServiceName	Specify the name of the target service.	Required
com.ibm.wsspi.wssecurity.krbtoken.targetServiceHost	Specify the host name that is associated with the target service in the following format: myhost.mycompany.com	Required
com.ibm.wsspi.wssecurity.krbtoken.targetServiceRealm	Specify the name of the realm that is associated with the target service.	Optional*

* If the targetServiceRealm property is not specified, the default realm name from the Kerberos configuration file is used as the realm name. To use Kerberos token security in a cross or trusted realm environment, you must provide a value for the targetServiceRealm property.

To specify multiple custom property name and value pairs, click **New**.

8) Click **Apply**.
9) From the Additional bindings heading, click **Callback handler**.
10) From the Class Name heading, select the **Use custom** option and specify com.ibm.websphere.wssecurity.callbackhandler.KRBTokenGenerateCallbackHandler in the associated field.
11) From the Basic Authentication heading, specify the appropriate values for the **User name**, **Password**, and **Confirm password** fields.

    The user name specifies the default user ID that is passed to the constructor of the callback handler; for example, kerberosuser.
12) Specify the token generator custom properties for Kerberos client principal name and password to initiate the Kerberos login.

    These custom properties control the prompt and establish the token based on the credential cache. Specify the following custom properties.

Table 286. Kerberos login custom properties. Use this property to specify the token generator information.

Name	Value	Type
com.ibm.wsspi.wssecurity.krbtoken.loginPrompt	Enables the Kerberos login when the value is True. The default value is False.	Optional

To specify multiple custom property name and value pairs, click **New**.

13) Click **Apply** and **OK**.

When you return to the Authentication and protection panel in the next step, a new protection token is defined for the token generator. To edit the configuration for this new token, click its name on the panel.

e. Optional: Return to the Authentication and protection panel to define a symmetric protection token for the token consumer. To return to the Authentication and protection panel, click the **Authentication and protection** link after the messages section of the panel.

Important: If you configure a symmetric protection token for the token consumer, ensure that you have previously defined a complimentary symmetric protection token for the token generator.

1) From the Protection tokens heading, click **New** and select **Token Consumer**.
2) Specify the name of the protection token in the **Name** field.
3) Select **Custom** from the values in the **Token type** menu list.
4) Specify the local name value in the **Local name** field.

   For interoperability with other web services technologies, specify the following local name: `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ`. If you are not concerned with interoperability issues, you can specify one of the following local name values:

   - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ`
   - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510`
   - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510`
   - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120`
   - `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120`

   These alternative values depend on the specification level for the Kerberos token that is generated by the Key Distribution Center (KDC). For more information about when to use these values, see Protection token settings (generator or consumer).
5) Do not specify a value for the **Namespace URI** field.
6) Select the **wss.consume.KRB5BST** value from the JAAS login drop-down menu.

   If you have previously defined your own Java Authentication and Authorization Service (JAAS) login module, you can select this login module to handle the Kerberos custom token. To define a custom JAAS login module, click **New Application Login** > **New**, specify an alias for the new module, and click **Apply**. For more information, see Login module settings for Java Authentication and Authorization Service.

   Attention: Although the information in the Login module settings for Java Authentication and Authorization Service topic refers to security and not Web Services Security, the configuration for a login module for Web Services Security is identical to security.
7) Click **Apply**.
8) From the Additional bindings heading, click **Callback handler**.

9) From the Class Name heading, select the **Use custom** option and specify `com.ibm.websphere.wssecurity.callbackhandler.KRBTokenConsumeCallbackHandler` in the associated field.

10) Click **Apply** and **OK**.

When you return to the Authentication and protection panel in the next step, you will see a new protection token defined for the token consumer. To edit the configuration for this new token, click its name on the panel.

f. Optional: Return to the Authentication and protection panel to define an authentication token configuration for the token generator. To return to the Authentication and protection panel, click the **Authentication and protection** link after the messages section of the panel.

Authentication tokens are sent in messages to prove or assert an identity.

Important: If you configure an authentication token for the token generator, you must define a complimentary authentication token for the token consumer.

1) From the Authentication tokens heading, click **New** and select **Token Generator**.

2) Specify the name of the authentication token in the **Name** field.

3) Select **Custom** from the values in the **Token type** menu list.

4) Specify the local name value in the **Local name** field.

For interoperability with other web services technologies, specify the following local name: `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ`. If you are not concerned with interoperability issues, you can specify one of the following local name values:

- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120`

These alternative values depend on the specification level for the Kerberos token that is generated by the Key Distribution Center (KDC). For more information about when to use these values, see Authentication generator or consumer token settings.

5) Do not specify a value for the **Namespace URI** field.

6) Select the **wss.generate.KRB5BST** value from the JAAS login menu list.

If you have previously defined your own Java Authentication and Authorization Service (JAAS) login module, you can select this login module to handle the Kerberos custom token. To define a custom JAAS login module, click **New Application Login** > **New**, specify an alias for the new module, and click **Apply**. For more information, see Login module settings for Java Authentication and Authorization Service.

Attention: Although the information in the Login module settings for Java Authentication and Authorization Service topic refers to security and not Web Services Security, the configuration for a login module for Web Services Security is identical to security.

7) Specify the token generator custom properties for the target service name, host, and realm.

The combination of the target service name and host values forms the Service Principal Name (SPN), which represents the target Kerberos service principal name. The Kerberos client requests the initial Kerberos AP_REQ token for the SPN. Specify the following custom properties.

Table 287. Target service custom properties. Use these custom properties to specify the token generator information.

Name	Value	Type
com.ibm.wsspi.wssecurity.krbtoken.targetServiceName	Specify the name of the target service.	Required
com.ibm.wsspi.wssecurity.krbtoken.targetServiceHost	Specify the host name that is associated with the target service in the following format: myhost.mycompany.com	Required
com.ibm.wsspi.wssecurity.krbtoken.targetServiceRealm	Specify the name of the realm that is associated with the target service.	Optional

To specify multiple custom property name and value pairs, click **New**.

8) Click **Apply**.
9) From the Additional bindings heading, click **Callback handler**.
10) From the Class Name heading, select the **Use custom** option and specify com.ibm.websphere.wssecurity.callbackhandler.KRBTokenGenerateCallbackHandler in the associated field.
11) From the Basic Authentication heading, specify the appropriate values for the **User name**, **Password**, and **Confirm password** fields.

    The user name specifies the default user ID that is passed to the constructor of the callback handler. For example: kerberosuser
12) Specify the token generator custom properties for Kerberos client principal name and password to initiate the Kerberos login.

    These custom properties control the prompt and establish the token based on the credential cache. Specify the following custom properties name and value pairs.

Table 288. Kerberos login custom properties. Use the custom properties to specify the token generator information.

Name	Value	Type
com.ibm.wsspi.wssecurity.krbtoken.loginPrompt	Enables the Kerberos login when the value is True. The default value is False.	Optional
com.ibm.wsspi.wssecurity.krbtoken.clientRealm	Specify the name of the Kerberos realm associated with the client	Optional*

* The clientRealm property is optional for a single Kerberos realm environment. When implementing Web Services Security in a cross or trusted Kerberos realm environment, you must provide a value for the clientRealm property.

If an application generates or consumes a Kerberos V5 AP_REQ token for each web services request message, set the com.ibm.wsspi.wssecurity.kerberos.attach.apreq custom property to **true** in the token generator and the token consumer bindings for the application

To specify multiple custom property name and value pairs, click **New**.

13) Click **Apply** and **OK**.

When you return to the Authentication and protection panel in the next step, you will see a new authentication token is defined for the token generator. To edit the configuration for this new token, click its name on the panel.

g. Optional: Return to the Authentication and protection panel to define an authentication token configuration for the token consumer. To return to the Authentication and protection panel, click the **Authentication and protection** link after the messages section of the panel.

Important: If you configure an authentication token for the token consumer, ensure that you have previously defined an authentication token for the token generator.

1) From the Authentication tokens heading, click **New** and select **Token Consumer**.
2) Specify the name of the authentication token in the **Name** field.
3) Select **Custom** from the values in the **Token type** menu list.
4) Specify the local name value in the **Local name** field.

For interoperability with other web services technologies, specify the following local name: `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ`. If you are not concerned with interoperability issues, you can specify one of the following local name values:

- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ1510`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ1510`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#Kerberosv5_AP_REQ4120`
- `http://docs.oasis-open.org/wss/oasis-wss-kerberos-token-profile-1.1#GSS_Kerberosv5_AP_REQ4120`

These alternative values depend on the specification level for the Kerberos token that is generated by the Key Distribution Center (KDC). For more information conditions under which to use these values, see the related link for the "Authentication generator or consumer token settings" topic.

5) Do not specify a value for the **Namespace URI** field.
6) Select the **wss.consume.KRB5BST** value from the JAAS login drop-down menu.

   If you have previously defined your own Java Authentication and Authorization Service (JAAS) login module, you can select this login module to handle the Kerberos custom token. To define a custom JAAS login module, click **New Application Login** > **New**, specify an alias for the new module, and click **Apply**. For more information, see Login module settings for Java Authentication and Authorization Service.

   Attention: Although the information in the Login module settings for Java Authentication and Authorization Service topic refers to security and not Web Services Security, the configuration for a login module for Web Services Security is identical to security.
7) Click **Apply**.
8) From the Additional bindings heading, click **Callback handler**.
9) From the Class Name heading, select the **Use custom** option and specify `com.ibm.websphere.wssecurity.callbackhandler.KRBTokenConsumeCallbackHandler` in the associated field.
10) Click **Apply** and **OK**.

When you return to the Authentication and protection panel in the next step, you will see a new authentication token is defined for the token consumer. To edit the configuration for this new token, click its name on the panel.

What to do next

You can optionally define key bindings for the request message protection and response message protection. If you choose to derive a key from the Kerberos token, configure the derived key information when you configure the key information for signature and encryption.

Return to the steps in the Configuring the Kerberos token for Web Services Security topic to ensure you have completed the steps for configuring the Kerberos token.

Updating the system JAAS login with the Kerberos login module:

Update the Kerberos system JAAS login module for JAX-WS applications.

About this task

If the Kerberos authentication mechanism is configured in the WebSphere Application Server security configuration for JAX-WS applications, the JAAS login wss.caller must be updated with the system JAAS login module for Kerberos. The login module is specified as com.ibm.ws.security.auth.kerberos.WSKrb5LoginModule.

There are two methods to update the Kerberos system JAAS login module: using the administrative console, or by running a Jython script.

Procedure

1. Using the administrative console, follow these steps:
   a. Click **Security > Global security > Java Authentication and Authorization Service > System logins**.
   b. Click on **wss.caller**, then click **New** to create a new JAAS login module.
   c. In the **Module class name** field, type `com.ibm.ws.security.auth.kerberos.WSKrb5LoginModule`.
   d. Click **OK**.
   e. In the wss.caller panel, click **Set Order**, then click on **WSKrb5LoginModule**.
   f. Move WSKrb5LoginModule up in the list of modules so that it is after com.ibm.ws.wssecurity.impl.auth.module.WSWSSLoginModule but before com.ibm.ws.security.server.lm.ltpaLoginModule. The order of the modules in the list is important. The finished list of modules should look like this:

```
com.ibm.ws.wssecurity.impl.auth.module.PreCallerLoginModule                      1
com.ibm.ws.wssecurity.impl.auth.module.UNTCallerLoginModule                      2
com.ibm.ws.wssecurity.impl.auth.module.X509CallerLoginModule                     3
com.ibm.ws.wssecurity.impl.auth.module.LTPACallerLoginModule                     4
com.ibm.ws.wssecurity.impl.auth.module.LTPAPropagationCallerLoginModule          5
com.ibm.ws.wssecurity.impl.auth.module.KRBCallerLoginModule                      6
com.ibm.ws.wssecurity.impl.auth.module.WSWSSLoginModule                          7
com.ibm.ws.security.auth.kerberos.WSKrb5LoginModule                           8
com.ibm.ws.security.server.lm.ltpaLoginModule                                    9
com.ibm.ws.security.server.lm.wsMapDefaultInboundLoginModule                     10
```

   g. Click **OK**, then click **Save** to save the changes.
   h. Restart the server.
2. You can also run a Jython script to update the module. For each cell, run the script **addKrbLoginModuleWSSCaller.py**, located in the *app_server_root*\bin directory, to update the WSKrb5LoginModule login module in the security configuration.
   a. Run the following command, where *app_server_root* is C:\WebSphere\AppServer:

```
wsadmin -conntype NONE -lang jython -f  C:\WebSphere\AppServer\bin\addKrbLoginModuleWSSCaller.py
```

   b. If the script is successful, the following message is displayed:

```
System JAAS login entry wss.caller has been updated.
```

   c. Restart the server.

Configuring Kerberos policy sets and V2 general sample bindings:

Configure the Kerberos policy sets and V2 general sample bindings that are included with WebSphere Application Server Version 7.0.0.1 and later.

Before you begin

In order to use the additional Kerberos policy sets and V2 general sample bindings that are included with the product, you must create a new profile after installing the product. Existing profiles are not automatically updated, and do not contain the Kerberos policy sets and V2 general sample bindings. You can update existing profiles manually using the following steps.

About this task

To update existing profiles, perform the following manual steps. The deployment manager profile and the stand-alone application server profile are the only profiles that you need to update.

Procedure

1. Copy the directories containing the additional Kerberos policy sets from the profile templates directory, *app_server_root*/profileTemplates/default/documents/config/templates/PolicySets, to the profile configuration directory, *profile_root*/config/templates/PolicySets. Each additional Kerberos policy set is contained in a separate directory. The directories are:
   - Kerberos V5 SecureConversation
   - Kerberos V5 WSSecurity default
   - TrustServiceKerberosDefault
2. Unpackage and copy the general bindings, Client sample V2 and Provider sample V2.
   a. Extract the directories and files from the package file *app_server_root*/profileTemplates/default/configArchives/AppSrv.car into a temporary directory.
   b. Copy the general binding directories from the temporary directory <*temp_dir*>/cells/defaultCell/bindings/, to the profile configuration directory for the cell, *profile_root*/config/cells/*<cellName>*/bindings. Each general binding is contained in a separate directory. The directories are:
      - Provider sample V2
      - Client sample V2
3. Restart the server.

Securing messages using SAML

Configure policy sets, bindings, and SAML-specific tokens to secure web services and messages.

About this task

To secure messages using SAML, you can import the SAML default policy sets and modify them to enable SAML function. Because WebSphere Application Server with SAML does not support attaching a policy set directly to a Web services client, you must specify the policy sets and bindings used to enable SAML as custom properties in the web services client binding document.

You can also create a SAML bearer token using the SAML library API. A bearer token contains a bearer assertion, which is used to facilitate web browser single sign-on (SSO). Other SAML set up tasks described in this section include configuring policy sets and bindings for a bearer token, or a holder-of-key token, or to communicate with a Security Token Service (STS).

See the following topics for more information about securing messages using SAML.

Signing SAML tokens at the message level:

Secure SAML tokens at the message level by enabling assertion signing.

Before you begin

Before configuring signing for SAML tokens, you must configure SAML policy sets and bindings to create SAML tokens as authentication supporting tokens, with message level integrity protection. For more information, read about securing messages using SAML. In addition, the attached SAML bindings must be application-specific bindings, not general bindings. The transform algorithm used for signing SAML assertions is different from other signed parts, while only one transform algorithm is used with general bindings.

About this task

To sign SAML assertions, a SOAP message must include a <wsse:SecurityTokenReference> element in the <wsse:Security> header block. The SecurityTokenReference (STR) is referenced by the message signature using a <ds:Reference> element. The security token reference must include a <wsse:KeyIdentifier> element with the ValueType value, http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID, or http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID, specifying the referenced assertion identifier. The <ds:Reference> element must include the URI of the STR-transform algorithm, http://docs.oasis-open.org/wss/2004/01/oasis-200401-wsssoap-message-security-1.0#STR-Transform. Use of STR-transform ensures that the SAML assertion itself is signed, not only the <wsse:SecurityTokenReference> element.

Follow these configuration steps to enable signing SAML tokens at the message level.

Procedure

1. Configure the message parts.
   a. From the administrative console, edit the SAML policy set, then click **WS-Security > Main policy > Request message part protection**.
   b. Select **Integrity protection**.
   c. Click **Add**.
   d. Enter a part name for **Name of part to be signed**; for example, saml_part.
   e. Under **Elements in Part**, click **Add**.
   f. Select **XPath**.
   g. Add two XPath expressions.

```
/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/'
and local-name()='Envelope']/*[namespace-uri()='http://schemas.xmlsoap.org/soap/envelope/'
and local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
and local-name()='Security']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
and local-name()='SecurityTokenReference']
```

```
/*[namespace-uri()='http://www.w3.org/2003/05/soap-envelope'
and local-name()='Envelope']/*[namespace-uri()='http://www.w3.org/2003/05/soap-envelope'
and local-name()='Header']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
and local-name()='Security']/*[namespace-uri()='http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd'
and local-name()='SecurityTokenReference']
```

   h. Click **Apply** and **Save**.
   i. Restart the application.
2. Configure protection and signing for the client.
   a. From the Service client policy set and bindings panel, click **WS-Security > Authentication and protection**.
   b. Under **Request message signature and encryption protection**, select a configured resource. The signature of the resource you select includes the SAML token.
      1) From the **Available** list under Message part reference, select the name of the part to be signed, as created in step 1; for example, saml_part.
      2) Click **Add**.
      3) In the **Assigned** list under Message part reference, highlight the name of the part you added; for example, saml_part.
      4) Click **Edit**.
      5) For the **Transform algorithms** setting, click **New**.
      6) Select **http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform**.
      7) Click **Apply**.
   c. Under Authentication tokens, select and edit the SAML token you want to sign.
      1) Under Custom property, click **New**.

2) Enter `signToken` as the custom property name.

   Note: The custom property is added at the token generator level, although it only applies to the SAML custom token. The property does not apply to other token types.

3) Enter `true` as the value of the custom property.
4) Click **Apply**.

3. Configure protection and signing for the service provider.
   a. From the Service provider policy sets and bindings panel, click **WS-Security > Authentication and protection**.
   b. Under **Request message signature and encryption protection**, select a configured resource. The signature of the resources you select includes the SAML token.
      1) From the **Available** list under Message part reference, select the name of the part to be signed, as created in step 1; for example, saml_part.
      2) Click **Add**.
      3) In the **Assigned** list under Message part reference, highlight the name of the part you added; for example, saml_part.
      4) Click **Edit**.
      5) For the **Transform algorithms** setting, click **New**.
      6) Select **http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform**.
      7) Click **Apply**.

Configuring policy sets and bindings to communicate with STS:

Configure policy sets and binding documents to enable a web services client to request SAML assertions from an external Security Token Service (STS).

Before you begin

This function is enabled in WebSphere Application Server Version 7.0.0.7 and later releases. After installing, you must create a new server profile, or add SAML configuration settings to an existing profile. Read about setting up the SAML configuration for more information.

About this task

WebSphere Application Server with SAML supports web services clients using the Web Services Security policy set and bindings when communicating with an external security token service (STS). In WebSphere Application Server Version 6.1 and later, web services clients use policy set and bindings to communicate with the target web services provider. A web services client uses two sets of policy set attachments: one set of policy set attachments for communicating to the target web services provider; and the other set of policy set attachments for communicating to the STS. Policy sets and bindings that are used when communicating with the target web services provider are attached to the web services client. In contrast, policy sets and bindings that enable STS communication are not directly attached to the web services clients. Instead, policy sets and bindings that enable STS communication are specified as custom properties in the web services client binding document. You can use general bindings or application-specific bindings to communicate with an STS. Using a general binding to access an STS is straightforward; simply specify the general binding name in the custom properties.

The procedure to configure application-specific bindings to access an STS is more involved. The administrative console is designed to manage policy set attachments to communicate with a web service provider. The console is not designed to manage a second set of policy set attachments to communicate to an STS. However, you can use the administrative console to manage a policy set attachment to access an STS, as described in the procedure.

Use the administrative console to attach the policy set that is used to access an STS to a web services client, and then create and modify an application-specific binding. Once the binding configuration is complete, detach the policy set and binding from the web services client. This procedure is necessary because the next step is to attach the policy set and bindings to communicate to the target web services provider. Detached application-specific bindings are not deleted from the file system, so the web services client bindings custom properties can successfully refer to the detached application-specific bindings.

The procedure uses a default application policy set, Username WSHTTPS default, as an example to describe the configuration steps to access the STS. The steps can also be applied to other policy sets. The web services application, JaxWSServicesSamples, is used in the example. JaxWSServicesSamples is not installed by default.

Procedure

1. Import the Username WSHTTPS default policy set. In this example, the Username WSHTTPS default policy is used to demonstrate the procedure, but you can use a different policy set to configure the bindings, if the policy set meets the policy requirements of the external STS.
   a. Click **Services > Policy sets > Application policy sets**.
   b. Click **Import**.
   c. Select **From Default Repository**.
   d. Select the **WSHTTPS default** policy set.
   e. Click **OK** to import the policy set.
2. Attach a policy set for the trust client. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings**. The steps which pertain to attaching and detaching the policy set, and configuring the trust client binding, are required only if an application-specific binding is used to access the external STS. You can skip these steps, and go to the step that discusses configuring communication with the STS, if you use a general binding to access the external STS.
   a. Select the check box for the web services client resource.
   b. Click **Attach Client Policy Set**.
   c. Select the policy set, **Username WSHTTPS default**.

   This step attaches the policy set to the web services trust client, as you would do to use this policy set for the application client to access the target web services. However, since you plan to use the Username WSHTTPS default policy set to access an external STS instead, the policy set is only temporarily attached to the Web services client. The purpose of this step is to allow you to use the administrative console to create or to modify the client binding document.
3. Configure the trust client binding.
   a. Select the web services client resource again.
   b. In the Service client policy sets and bindings panel, click **Assign Binding**.
   c. Click **New Application Specific Binding** to create an application-specific binding.
   d. Specify a binding configuration name for the new application-specific binding. In this example, the binding name is **SamlTCSample**.
   e. Add the **SSL transport** policy type to the binding. Optionally, you can modify the NodeDefaultSSLSettings settings. Click **Security > SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings**.
4. Optional: You can create an HTTP transport binding using the previous steps if you want to configure a user name and password to add to the HTTP header, or if you want to configure a proxy. If you elect not to create an HTTP transport binding, the web services runtime environment uses the default HTTP transport settings.
5. Add the **WS-Security** policy type to the binding, then modify the authentication settings.

a. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings > *SamlTCSample* > Add > WS-Security > Authentication and protection > request:uname_token**.
b. Click **Apply**.
c. Select **Callback handler**.
d. Specify a user name and password (and confirm the password) to authenticate the web services client to the external STS.
e. Click **OK** and **Save**.

6. After the binding settings are saved, return to the Service client policy sets and bindings panel to detach the policy set and bindings.
   a. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings**.
   b. Click the check box for the web services client resource.
   c. Click **Detach client policy set**.

   The application-specific binding configuration you created in the previous steps is not deleted from the file system when the policy set is detached. This means that you can still use the application-specific binding you created to access the STS.
7. Import the SSL certificate from the external STS.
   a. Click **Security > SSL certificate and key management > Manage endpoint security configurations > *server_or_node_endpoint* > Keystores and certificates > NodeDefaultTrustStore > Signer certificates**.
   b. Click **Retrieve from port**.
   c. Specify the host name and port number of the external STS server, and assign an alias to the certificate. Use the SSL STS port.
   d. Click **Retrieve signer information**.
   e. Click **Apply** and **Save** to copy the retrieved certificate to the NodeDefaultTrustStore object.
8. Optional: If further modifications to the wstrustClientBinding configuration are needed, and the wstrustClientBinding property is pointing to an application-specific binding, you must attach the application-specific binding to the web services client before you can complete the modifications. The attachment is temporary. As detailed in the previous steps, you can detach the modified application-specific binding from the web service client after the modification is completed.

Results

After successfully completing the steps, the web services client is ready to send requests to the external STS. To enable this function, the following conditions and settings were activated when you completed the procedure:

- Attach a policy set and bindings that propagate SAML tokens. For example, attach the SAML11 Bearer WSHTTPS default policy set and the Saml Bearer Client sample general binding to the web service client.
- The Username WSHTTPS default policy set and an application-specific binding, SamlTCSample, are referenced in the Saml Bearer Client sample binding, and are set up to access the external STS.
- The external STS SSL certificate has been retrieved and added to the NodeDefaultTrustStore truststore.
- To confirm that you successfully enabled the function, you can configure the trace setting, com.ibm.ws.wssecurity.*=all=enabled. The trace shows that SAML assertions are issued by the external STS, for example:

```
[8/23/09 18:26:59:252 CDT] 0000001f TrustSecurity 3   Security Token Service reponse:
[8/23/09 18:26:59:392 CDT] 0000001f TrustSecurity 3
<?xml version="1.0" encoding="UTF-8"?><s:Envelope xmlns:s="http://www.w3.org/2003/05/soap-envelope"
   xmlns:a="http://www.w3.org/2005/08/addressing" xmlns:u="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
    <s:Header>
        <a:Action s:mustUnderstand="1">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</a:Action>
        <a:RelatesTo>urn:uuid:663A7B27BA8EB2CF9D12510700299 34</a:RelatesTo>
        <o:Security xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" s:mustUnderstand="1">
```

```
            <u:Timestamp u:Id="_0">
                <u:Created>2009-08-23T23:26:57.664Z</u:Created>
                <u:Expires>2009-08-23T23:31:57.664Z</u:Expires>
            </u:Timestamp>
        </o:Security>
    </s:Header>
    <s:Body>
      <trust:RequestSecurityTokenResponseCollection xmlns:trust="http://docs.oasis-open.org/ws-sx/ws-trust/200512">
        <trust:RequestSecurityTokenResponse>
          <trust:Lifetime>
            <wsu:Created
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-08-23T23:26:57.648Z</wsu:Created>
            <wsu:Expires
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2009-08-24T09:26:57.648Z</wsu:Expires>
          </trust:Lifetime>
        <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
          <a:EndpointReference>
             <a:Address>https://taishan.austin.ibm.com:9443/WSSampleSei/EchoService12</a:Address>
           </a:EndpointReference>
        </wsp:AppliesTo>
        <trust:RequestedSecurityToken>
          <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion" MajorVersion="1" MinorVersion="1"
AssertionID="_3c656382-9916-4e5f-9a16-fe0287dfc409" Issuer="http://svt193.svt193domain.com/Trust" IssueInstant="2009-08-23T23:26:57.663Z">
          <saml:Conditions NotBefore="2009-08-23T23:26:57.648Z" NotOnOrAfter="2009-08-24T09:26:57.648Z">
          <saml:AudienceRestrictionCondition>
          <saml:Audience>https://taishan.austin.ibm.com:9443/WSSampleSei/EchoService12</saml:Audience>
          </saml:AudienceRestrictionCondition>
          </saml:Conditions>
          <saml:AuthenticationStatement AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password"
AuthenticationInstant="2009-08-23T23:26:57.640Z">
          <saml:Subject>
          <saml:SubjectConfirmation>
          <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:ConfirmationMethod>
          </saml:SubjectConfirmation>
          </saml:Subject>
          </saml:AuthenticationStatement>
            <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
            <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
            <ds:Reference URI="#_3c656382-9916-4e5f-9a16-fe0287dfc409">
            <ds:Transforms>
            <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
            <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>YGySZX4VPv25R+oyzFpE0/T/tjs=</ds:DigestValue>
            </ds:Reference>
            </ds:SignedInfo>
                 <ds:SignatureValue>eP68...Vr08=</ds:SignatureValue>
                   <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
                      <X509Data>
                        <X509Certificate>MII...ymqg3</X509Certificate>
                      </X509Data>
                    </KeyInfo>
                  </ds:Signature>
           </saml:Assertion>
        </trust:RequestedSecurityToken>
       <trust:RequestedAttachedReference>
          <o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <o:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID">_3c656382-9916-4e5f-9a16-fe0287dfc409</o:KeyIdentifier>
          </o:SecurityTokenReference>
        </trust:RequestedAttachedReference>
        <trust:RequestedUnattachedReference>
          <o:SecurityTokenReference xmlns:o="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
            <o:KeyIdentifier
ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.0#SAMLAssertionID">_3c656382-9916-4e5f-9a16-fe0287dfc409</o:KeyIdentifier>
          </o:SecurityTokenReference>
        </trust:RequestedUnattachedReference>
        <trust:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1</trust:TokenType>
        <trust:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</trust:RequestType>
        <trust:KeyType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer</trust:KeyType>
        </trust:RequestSecurityTokenResponse>
        </trust:RequestSecurityTokenResponseCollection>
    </s:Body>
</s:Envelope>
```

What to do next

Complete the web service client and web service provider configuration. Read about configuring client and provider bindings for the SAML bearer token for more information.

Configuring client and provider bindings for the SAML bearer token:

Configure the client and provider policy set attachments and bindings for the SAML bearer token.

Before you begin

This function is enabled in WebSphere Application Server Version 7.0.0.7 and later releases. After installing, you must create one or more new server profiles, or add SAML configuration settings to an existing profile. For example, in a network deployment environment, there are multiple profiles. Read about

setting up the SAML configuration for more information.

About this task

A SAML bearer token is a SAML token that uses the Bearer subject confirmation method. In a bearer subject confirmation method, a sender of SOAP messages is not required to establish correspondence that binds a SAML token with contents of the containing SOAP message.

WebSphere Application Server with SAML provides numerous default SAML token application policy sets and several general client and provider binding samples. Before you can configure the client and provider bindings for the SAML bearer token, you must import one of these default policy sets: SAML20 Bearer WSHTTPS default, SAML20 Bearer WSSecurity default, SAML11 Bearer WSHTTPS default, or SAML11 Bearer WSSecurity default. The SAML11 policy sets are almost identical to the SAML20 policy sets, except that the SAML20 policy sets support the SAML Version 2.0 token type, while the SAML11 policy sets support the Version 1.1 token type.

A SAML token policy is defined by a CustomToken extension in the application server. To create the CustomToken extension, define the SAML token configuration parameters in terms of custom properties in the client and provider binding document. The Saml Bearer Client sample and the Saml Bearer Provider sample general bindings contain the essential configuration for the custom properties. The client and provider sample bindings contain both SAML11 and SAML20 token type configuration information and therefore can be used with both SAML11 and SAML20 policy sets. Depending on how you plan to implement the SAML tokens, you must modify the property values in the installed binding samples. Examples of the properties and property values are provided in the procedure.

The procedure for modifying the binding sample begins with configuring the web services client policy set attachment, and then modifies the web services provider policy set attachment. The example presented in the procedure uses the sample web services application JaxWSServicesSamples.

Procedure

1. Import two default policy sets: the Username WSHTTPS default policy set, and one of the following bearer policy sets:
   - SAML11 Bearer WSHTTPS default
   - SAML20 Bearer WSHTTPS default
   - SAML20 Bearer WSSecurity default
   - SAML11 Bearer WSSecurity default

   a. Click **Services** > **Policy sets** > **Application policy sets**.
   b. Click **Import**.
   c. Select **From Default Repository**.
   d. Select the two default policy sets.
   e. Click **OK** to import the policy sets.
2. Attach a policy set for the trust client. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings**. The steps which pertain to attaching and detaching the policy set, and configuring the trust client binding, are required only if an application-specific binding is used to access the external STS. You can skip these steps, and go to the step that discusses configuring communication with the STS, if you use a general binding to access the external STS.
   a. Select the check box for the web services client resource.
   b. Click **Attach Client Policy Set**.
   c. Select the policy set, **Username WSHTTPS default**.

   This step attaches the policy set to the web services trust client, as you would do to use this policy set for the application client to access the target web services. However, since you plan to use the

Username WSHTTPS default policy set to access an external STS instead, the policy set is only temporarily attached to the Web services client. The purpose of this step is to allow you to use the administrative console to create or to modify the client binding document.

3. Configure the trust client binding.
   a. Select the web services client resource again.
   b. In the Service client policy sets and bindings panel, click **Assign Binding**.
   c. Click **New Application Specific Binding** to create an application-specific binding.
   d. Specify a binding configuration name for the new application-specific binding. In this example, the binding name is **SamlTCSample**.
   e. Add the **SSL transport** policy type to the binding. Optionally, you can modify the NodeDefaultSSLSettings settings. Click **Security > SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings**.
4. Add the **WS-Security** policy type to the binding, then modify the authentication settings.
   a. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings > *SamlTCSample* > Add > WS-Security > Authentication and protection > request:uname_token**.
   b. Click **Apply**.
   c. Select **Callback handler**.
   d. Specify a user name and password (and confirm the password) to authenticate the web services client to the external STS.
   e. Click **OK** and **Save**.
5. After the binding settings are saved, return to the Service client policy sets and bindings panel to detach the policy set and bindings.
   a. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings**.
   b. Click the check box for the web services client resource.
   c. Click **Detach client policy set**.

   The application-specific binding configuration you created in the previous steps is not deleted from the file system when the policy set is detached. This means that you can still use the application-specific binding you created to access the STS.
6. Configure the STS endpoint URL and the username and password to authenticate to the STS.
   a. Click **Applications** > **Application types** > **WebSphere enterprise applications** > ***JaxWSServicesSamples*** > **Service client policy sets and bindings** > **Saml Bearer Client sample** > **WS-Security** > **Authentication and protection**.
   b. Click **gen_saml11token** in the Authentication tokens table.
   c. Click **Callback handler**.
   d. Modify the **stsURI** property to specify the STS endpoint. For the self-issuer in an intermediate server, this property is not required, or if the property is specified, it is set to `www.websphere.ibm.com/SAML/Issuer/Self`.
   e. Verify that the value of the **confirmationMethod** property is **bearer**, or modify the property value if necessary.
   f. Verify that the value of the **keyType** property is **http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer**, or modify the property value if necessary.
   g. Verify that the value of the **wstrustClientPolicy** property is **Username WSHTTPS default**, or modify the property if necessary.
   h. Modify the **wstrustClientBinding** property and change the value to match the application-specific binding created in the previous steps. For this example, the value is **SamlTCSample**.
   i. Optional: To change how the application server searches for the binding, you can specify the **wstrustClientBindingScope** property and set its value to either **application** or **domain**. When the

value is set to domain, the application server searches for the wstrustClientBinding at the file system location that contains general binding documents. When the value is set to application, the application server searches for the wstrustClientBinding at the file system location that contains application-specific binding documents. When the wstrustClientBindingScope property is not specified, the default behavior of the application server is to search for application-specific bindings and then search for general bindings. If the wstrustClientBinding cannot be located, the application server uses the default bindings.

j. Optional: You can modify the default trust client SOAP version, which is the same as the application client. Set the custom property **wstrustClientSoapVersion** to the value `1.1` to change to SOAP Version 1.1, or set the property to the value `1.2` to change to SOAP Version 1.2.

k. Click **Apply** and **Save**.

7. Optional: If further modifications to the wstrustClientBinding configuration are needed, and the wstrustClientBinding property is pointing to an application-specific binding, you must attach the application-specific binding to the web services client before you can complete the modifications. The attachment is temporary. As detailed in the previous steps, you can detach the modified application-specific binding from the web service client after the modification is completed.
8. Import the SSL certificate from the external STS.
   a. Click **Security > SSL certificate and key management > Manage endpoint security configurations > *server_or_node_endpoint* > Keystores and certificates > NodeDefaultTrustStore > Signer certificates**.
   b. Click **Retrieve from port**.
   c. Specify the host name and port number of the external STS server, and assign an alias to the certificate. Use the SSL STS port.
   d. Click **Retrieve signer information**.
   e. Click **Apply** and **Save** to copy the retrieved certificate to the NodeDefaultTrustStore object.
9. Restart the web services client application so that the policy set attachment modifications can take effect.
10. Attach the **SAML11 Bearer WSHTTPS default** policy set to the web services provider.
11. Assign the **Saml Bearer Provider sample** general binding.
12. Click **WebSphere enterprise applications** > ***JaxWSServicesSamples*** > **Service provider policy sets and bindings** > **Saml Bearer Provider sample** > **WS-Security** > **Authentication and protection**.
    a. Click **con_saml11token** in the Authentication tokens table.
    b. Click the **Callback handler** link.
    c. Use this panel to configure the SAML token issuer digital signature validation binding data for authentication to the external STS, as described in the following step.
13. Add the external STS signing certificate to the truststore for the provider. This step is required if the SAML assertions are signed by the STS and the **signatureRequired** custom property is not specified, or has a value of `true`.

    A trust store specifies keystores that contain trusted root certificates that validate the signer certificate of the SAML token. The SAML token consumer uses this keystore to validate the SAML signer certificate of the digital signature.
    a. Set the custom property **trustStoreType** to match the key store type. Supported keys store types include: **jks**, **jceks**, and **pkcs12**.
    b. Set the custom property **trustStorePath** to the key store file location. For example, *app_server_root*`/etc/ws-security/samples/dsig-issuer.jceks`. The file dsig_issuer.jceks is not provided when WebSphere Application Server is installed, so you must create the file.
    c. Set the custom property **trustStorePassword** to the value of the truststore password. The password is stored as a custom property and is encoded by the administrative console.

d. Optional: You can set the custom property **trustedAlias** to a value such as `samlissuer`. If this property is specified, the X.509 certificate represented by the alias is the only STS certificate that is trusted for SAML signature verification. If this custom property is not specified, the web services runtime environment uses the signing certificate inside the SAML assertions to validate the SAML signature and then verifies the certificate against the configured truststore.

e. Optional: You can set the custom property **trustAnySigner** to the value `true` to allow no signer certificate validation. The Trust Any certificate configuration setting is ignored for the purposes of SAML signature validation.

f. Optional: You can set the custom property **signatureRequired** to `false`, which waives digital signature validation. However, a good security practice is to require SAML assertions to be signed and always require issuer digital signature validation.

g. If the SAML assertion is encrypted by the STS with the PublicKey from the recipient, you must configure the private key from the recipient to decrypt the EncryptedAssertion. The following callback handler custom properties must be set to the value described in the table for the recipient to decrypt the SAML assertion:

Custom property	Value
keyStorePath	Keystore location
keyStoreType	Matching keystore type Supported keystore types include: jks, jceks, and pkcs12
keyStorePassword	Password for the keystore
keyAlias	The alias of the public key used for SAML encryption
keyName	The name of the public key used for SAML encryption
keyPassword	The password for the key name

h. Optional: You can configure the recipient to validate either the issuer name or the certificate SubjectDN of the issuer in the SAML assertion, or you can validate both. Create a list of trusted issuer names, or a list of trusted certificate SubjectDNs, or both types of lists. If you create both issuer name and SubjectDN lists, both issuer name and SubjectDN are verified. If the received SAML issuer name or signer SubjectDN is not in the trusted list, SAML validation fails, and an exception is issued. This example shows how to create a list of trusted issuers and trusted SubjectDNs.

For each trusted issuer name, use *trustedIssuer_n* where *n* is a positive integer. For each trusted SubjectDN, use *trustedSubjectDN_n* where *n* is a positive integer. If you create both types of lists, the integer *n* must match in both lists for the same SAML assertion. The integer *n* starts with 1, and increments by 1.

In this example, you trust a SAML assertion with the issuer name `WebSphere/samlissuer`, regardless of the SubjectDN of the signer, so you add the following custom property:

```
<properties value="WebSphere/samlissuer" name="trustedIssuer_1"/>
```

In addition, you trust a SAML assertion issued by `IBM/samlissuer`, when the SubjectDN of the signer is `ou=websphere,o=ibm,c=us`, so you add the following custom properties:

```
<properties value="IBM/samlissuer" name="trustedIssuer_2"/>
<properties value="ou=websphere,o=ibm,c=us" name="trustedSubjectDN_2"/>
```

By default, WebSphere Application Server trusts all SAML issuers when you do not define a `trustedIssuer_n` value. Without knowing this default behavior, you might mistakenly accept SAML assertions that are issued by an unauthorized STS

i. Optional: You can add a list of non-root certificate authority (CA) certificates that can be used to check the signature of the SAML token. To add non-root certificates, add a custom property named `X509PATH_n` where *n* is a non-negative integer as the value for the non-root certificates.

j. Optional: You can add a list of certificate revocation lists (CRLs) that can be used to validate the signature of the SAML token. To add CRLs, add a custom property named `CRLPATH_n` where *n* is a non-negative integer as the value for the CRLs.

k. Click **Apply** and **Save**.

14. Verify that the token is configured as a bearer token.
    a. Verify that the value of the **confirmationMethod** is `Bearer`.
    b. Verify that the value of the **keyType** property value is `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer` or the `bearer` alias. The `wstrustClientWSTNamespace` property determines how the `bearer` alias is interpreted. In this case, it is assumed that it is set to the WS-Trust 1.3 namespace. If it had a value of WS-Trust 1.2, the `bearer` alias carries no meaning, as it is not defined for WS-Trust 1.2.
    c. Click **Apply** and **Save**.
15. Optional: You can configure the caller binding to select a SAML token to represent the requester identity. The Web Services Security runtime environment uses the specified JAAS login configuration to acquire the user security name and group membership data from the user registry using the SAML token NameId or NameIdentifier as the user name.
    a. Click **WebSphere enterprise applications** > ***JaxWSServicesSamples*** > **Service provider policy sets and bindings** > **Saml Bearer Provider sample** > **WS-Security** > **Callers**.
    b. Click **New** to create the caller configuration
    c. Specify a **Name**, such as `caller`.
    d. Enter a value for the **Caller identity local part**. For example, `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV1.1`, which is the local part of the CustomToken element in the attached WS-Security policy.
    e. Click **Apply** and **Save**.
16. Restart the web services provider application so that the policy set attachment modifications can take effect.

Results

When you have completed the procedure, the JaxWSServicesSamples web services application is ready to use the SAML Bearer default policy set, the Saml Bearer Client sample, and the Saml Bearer Provider sample general bindings.

Configuring client and provider bindings for the SAML holder-of-key symmetric key token:

Configure the client and provider policy set attachments and bindings for the SAML holder-of-key token. This configuration scenario uses a symmetric key.

Before you begin

This function is enabled in WebSphere Application Server Version 7.0.0.7 and later releases. After installing, you must create one or more new server profiles, or add SAML configuration settings to an existing profile. For example, in a network deployment environment, there are multiple profiles. Read about setting up the SAML configuration for more information.

About this task

WebSphere Application Server with SAML provides numerous default SAML token application policy sets and several general client and provider binding samples. Before you can configure the client and provider bindings for the SAML holder-of-key token, you must import one of these default policy sets: SAML20 HoK Symmetric WSSecurity default or SAML11 HoK Symmetric WSSecurity default. The SAML11 policy sets are almost identical to the SAML20 policy sets, except that SAML20 HoK Symmetic WSSecurity default

policy set supports the SAML Version 2.0 token type, while the SAML11 HoK Symmetric WSSecurity default policy set supports the Version 1.1 token type.

The SAML token policy is defined by a CustomToken extension in the application server. To create the CustomToken extension, define the SAML token configuration parameters in terms of custom properties in the client and provider binding document. The Saml HoK Symmetric Client sample and the Saml HoK Symmetric Provider sample general bindings contain the essential configuration for the custom properties. The client and provider sample bindings contain both SAML11 and SAML20 token type configuration information and therefore can be used with both SAML11 and SAML20 policy sets. Depending on how you plan to implement the SAML tokens, you must modify the property values in the installed binding samples. Examples of the properties and property values are provided in the procedure.

The procedure for modifying the binding sample begins with configuring the web services client policy set attachment, then configuring the web services provider policy set attachment. The example presented in the procedure uses the sample web services application JaxWSServicesSamples.

Procedure

1. Import two default policy sets: SAML20 HoK Symmetric WSSecurity default, and the Username WSHTTPS default.
   a. Click **Services** > **Policy sets** > **Application policy sets**.
   b. Click **Import**.
   c. Select **From Default Repository**.
   d. Select the two default policy sets.
   e. Click **OK** to import the policy sets.

   If you do not want the server to automatically request a SAML token from the Security Token Service (STS) using the WS-Trust client, you can skip steps 2, 3, and 4, and continue to step 5. For example, you can skip steps 2, 3, and 4, if web services act as a client and self-issues a SAML token based on the original SAML token, or the web services client has already acquired a SAML token and cached the SAML token in the RequestContext.
2. Attach a policy set for the trust client. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings**. The steps which pertain to attaching and detaching the policy set, and configuring the trust client binding, are required only if an application-specific binding is used to access the external STS. You can skip these steps, and go to the step that discusses configuring communication with the STS, if you use a general binding to access the external STS.
   a. Select the check box for the web services client resource.
   b. Click **Attach Client Policy Set**.
   c. Select the policy set, **Username WSHTTPS default**.

   This step attaches the policy set to the web services trust client, as you would do to use this policy set for the application client to access the target web services. However, since you plan to use the Username WSHTTPS default policy set to access an external STS instead, the policy set is only temporarily attached to the Web services client. The purpose of this step is to allow you to use the administrative console to create or to modify the client binding document.
3. Configure the trust client binding.
   a. Select the web services client resource again.
   b. In the Service client policy sets and bindings panel, click **Assign Binding**.
   c. Click **New Application Specific Binding** to create an application-specific binding.
   d. Specify a binding configuration name for the new application-specific binding. In this example, the binding name is **SamlTCSample**.

e. Add the **SSL transport** policy type to the binding. Optionally, you can modify the NodeDefaultSSLSettings settings. Click **Security > SSL certificate and key management > SSL configurations > NodeDefaultSSLSettings**.

4. Add the **WS-Security** policy type to the binding, then modify the authentication settings.
   a. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings > *SamlTCSample* > Add > WS-Security > Authentication and protection > request:uname_token**.
   b. Click **Apply**.
   c. Select **Callback handler**.
   d. Specify a user name and password (and confirm the password) to authenticate the web services client to the external STS.
   e. Click **OK** and **Save**.
5. After the binding settings are saved, return to the Service client policy sets and bindings panel to detach the policy set and bindings.
   a. Click **Applications > Application types > WebSphere enterprise applications > *JaxWSServicesSamples* > Service client policy sets and bindings**.
   b. Click the check box for the web services client resource.
   c. Click **Detach client policy set**.

   The application-specific binding configuration you created in the previous steps is not deleted from the file system when the policy set is detached. This means that you can still use the application-specific binding you created to access the STS.
6. Download the unrestricted jurisdiction policy file. The SAML20 HoK Symmetric WSSecurity default security policy uses the 256 bit encryption key size, which requires the unrestricted Java Cryptography Extension (JCE) policy file. For more information, read the section Using the unrestricted JCE policy files in the Tuning Web Services Security topic.
7. Attach the SAML20 HoK Symmetric WSSecurity default policy set and assign the Saml HoK Symmetric Client sample binding to the client resource.
   a. Click **Applications** > **Application types** > **WebSphere enterprise applications** > ***JaxWSServicesSamples*** > **Service client policy sets and bindings**.
   b. Select the web services client resource.
   c. Click **Attach Client Policy Set**.
   d. Select the policy set, **SAML20 HoK Symmetric WSSecurity default**.
   e. Select the web services client resource again.
   f. In the Service client policy sets and bindings panel, click **Assign Binding**.
   g. Select the **Saml HoK Symmetric Client sample** general binding.
   h. Click **Save**.
8. Configure the STS endpoint URL and the user name and password to authenticate to the STS.
   a. Click **Applications** > **Application types** > **WebSphere enterprise applications** > ***JaxWSServicesSamples*** > **Service client policy sets and bindings** > **Saml HoK Symmetric Client sample** > **WS-Security** > **Authentication and protection**.
   b. Click **gen_saml20token** in the Protection tokens table.
   c. Click **Callback handler**.
   d. Modify the **stsURI** property and specify the STS endpoint. If you do not use an external STS, and you want the application server to self-issue a holder-of-key assertion with a symmetric key, do not complete this step and go to step 8i.
   e. If necessary, modify the **wstrustClientPolicy** property and change the value to **Username WSHTTPS default**.
   f. Modify the **wstrustClientBinding** property and change the value to match the application-specific binding created in the previous steps. For this example, the value is **SamlTCSample**. This step

attaches the WS-Trust client policy set. You can skip this step if you do not want the server to automatically request a SAML token from the STS using the WS-Trust client.

g. Change the value of the **wstrustClientBindingScope** property, which controls how the application server searches for the binding. Set the property value to either **application** or **domain**. When the value is set to domain, the application server searches for the wstrustClientBinding at the file system location that contains general binding documents. When the value is set to application, the application server searches for the wstrustClientBinding at the file system location that contains application-specific binding documents. When the wstrustClientBindingScope property is not specified, the default behavior of the application server is to search for application-specific bindings and then search for general bindings. If the wstrustClientBinding can not be located, the application server uses the default bindings.

h. Verify that the value of the `confirmationMethod` property is `Holder-of-key`.

i. Verify that the value of the `keyType` property value is `http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey` or the `symmetrickey` alias. The `wstrustClientWSTNamespace` property determines how the `symmetrickey` alias is interpreted. In this case it is assumed that it is set to the WS-Trust 1.3 namespace. If it had a value of WS-Trust 1.2, the `symmetrickey` alias is interpreted as `http://schemas.xmlsoap.org/ws/2005/02/trust/SymmetricKey`.

j. Optional: You can modify the default trust client SOAP version, which is the same as the application client. Set the custom property **wstrustClientSoapVersion** to the value `1.1` to change to SOAP Version 1.1, or set the property to the value `1.2` to change to SOAP Version 1.2.

k. Optional: If you are not using an external STS, and you want the application server to self-issue a holder-of-key assertion with a symmetric key, set the custom property **recipientAlias** to the value of the key alias of the target service. Specifying this property protects the symmetric key for the target service. This alias must be a valid key alias that is contained in the configured trust store of the SAML issuer. The TrustStorePath property specifies the location of the trust store file. The TrustStorePath property is defined in the `SAMLIssuerConfig.properties` file for the application server. For example, the location of the `SAMLIssuerConfig.properties` file at the server level on a WebSphere Application server is:

app_server_root`/profiles/$PROFILE/config/cells/$CELLNAME/nodes/$NODENAME/servers/$SERVERNAME/SAMLIssuerConfig.properties`

The location of this file at the cell level on a WebSphere Application server is:

app_server_root`/profiles/$PROFILE/config/cells/$CELLNAME/sts/SAMLIssuerConfig.properties`

l. Click **Apply** and **Save**.

9. Optional: If further modifications to the wstrustClientBinding configuration are needed, and the wstrustClientBinding property is pointing to an application-specific binding, you must attach the application-specific binding to the web services client before you can complete the modifications. The attachment is temporary. As detailed in the previous steps, you can detach the modified application-specific binding from the web service client after the modification is completed.
10. Import the SSL certificate from the external STS.
    a. Click **Security > SSL certificate and key management > Manage endpoint security configurations > *server_or_node_endpoint* > Keystores and certificates > NodeDefaultTrustStore > Signer certificates**.
    b. Click **Retrieve from port**.
    c. Specify the host name and port number of the external STS server, and assign an alias to the certificate. Use the SSL STS port.
    d. Click **Retrieve signer information**.
    e. Click **Apply** and **Save** to copy the retrieved certificate to the NodeDefaultTrustStore object.
11. Restart the web services client application so that the policy set attachment modifications can take effect.
12. Attach the **SAML20 HoK Symmetric WSSecurity default** policy set to the web services provider.
13. Download the unrestricted jurisdiction policy file. The SAML20 HoK Symmetric WSSecurity default security policy uses the 256 bit encryption key size, which requires the unrestricted Java

Cryptography Extension (JCE) policy file. For more information, read the section Using the unrestricted JCE policy files in the Tuning Web Services Security applications topic.

14. Assign the **Saml HoK Symmetric Provider sample** general binding.
15. Click **Applications** > **Application types** > **WebSphere enterprise applications** > ***JaxWSServicesSamples*** > **Service provider policy sets and bindings** > **Saml HoK Symmetric Provider sample** > **WS-Security** > **Authentication and protection**.
    a. Click **con_saml20token** in the Authentication tokens table.
    b. Click the **Callback handler** link.
    c. Use this panel to configure the embedded symmetric key decryption configuration, and the SAML token issuer digital signature validation to the external STS, as described in the following step.
16. Configure the binding data to decrypt the embedded secret key, or the SAML assertion that is protected by the public key from the recipient. The STS must have access to the public key of the recipient. There are two options to configure the keys for decryption:
    - Option 1: Configure the keystore and a private key, as follows:
      a. Verify that the **Keystore name** field has the value `custom`.
      b. Click **Custom keystore configuration** to view and edit the keystore configuration.
      c. Verify that the initial value for the key file is *app_server_root*`/etc/ws-security/samples/enc-service.jceks`.
    - Option 2: Set the custom properties in the callback handler as follows:

Custom property	Value
keyStorePath	Keystore location
keyStoreType	Matching keystore type Supported keystore types include: jks, jceks, and pkcs12
keyStorePassword	Password for the keystore
keyAlias	The alias of the public key used for SAML encryption
keyName	The name of the public key used for SAML encryption
keyPassword	The password for the key name

17. Add the external STS signing certificate to the truststore. This step is required if the SAML assertions are signed by the STS and the **signatureRequired** custom property is not specified, or has a value of **true**. This truststore is configured for the service provider.
    a. Set the custom property **trustStoreType** to match the keystore type. Supported keystore types include: **jks**, **jceks**, and **pkcs12**.
    b. Set the custom property **trustStorePath** to the keystore file location. For example, *app_server_root*`/etc/ws-security/samples/dsig-issuer.jceks`. The file dsig_issuer.jceks is not provided when WebSphere Application Server is installed, so you must create the file.
    c. Set the custom property **trustStorePassword** to the encoded value of the store password. The password is stored as a custom property and is encoded by the administrative console.
    d. Optional: You can set the custom property **trustedAlias** to a value such as `samlissuer`. Do not set the trustedAlias property if the SAML token is signed by different signers, for example, if the STS delegates token requests to different token providers, and each provider signs with a certificate. If this custom property is not specified, the web services runtime environment uses the signing certificate password in the SAML assertions to validate the signature and then verifies the certificate against the configured truststore.
    e. Optional: You can set the custom property **trustAnySigner** to the value `true` to allow no signer certificate validation. The Trust Any certificate configuration setting is ignored for the purposes of SAML signature validation.

f. Optional: You can set the custom property **signatureRequired** to `false`, which waives digital signature validation. However, a good security practice is to require SAML assertions to be signed and always require issuer digital signature validation.

g. Optional: You can configure the recipient to validate either the issuer name or the certificate SubjectDN of the issuer in the SAML assertion, or you can validate both. Create a list of trusted issuer names, or a list of trusted certificate SubjectDNs, or both types of lists. If you create both issuer name and SubjectDN lists, both issuer name and SubjectDN are verified. If the received SAML issuer name or signer SubjectDN is not in the trusted list, SAML validation fails, and an exception is issued. This example shows how to create a list of trusted issuers and trusted SubjectDNs.

   For each trusted issuer name, use *trustedIssuer_n* where *n* is a positive integer. For each trusted SubjectDN, use *trustedSubjectDN_n* where *n* is a positive integer. If you create both types of lists, the integer *n* must match in both lists for the same SAML assertion. The integer *n* starts with 1, and increments by 1.

   In this example, you trust a SAML assertion with the issuer name `WebSphere/samlissuer`, regardless of the SubjectDN of the signer, so you add the following custom property:

```
<properties value="WebSphere/samlissuer" name="trustedIssuer_1"/>
```

   In addition, you trust a SAML assertion issued by `IBM/samlissuer`, when the SubjectDN of the signer is `ou=websphere,o=ibm,c=us`, so you add the following custom properties:

```
<properties value="IBM/samlissuer" name="trustedIssuer_2"/>
<properties value="ou=websphere,o=ibm,c=us" name="trustedSubjectDN_2"/>
```

   By default, WebSphere Application Server trusts all SAML issuers when you do not define a `trustedIssuer_n` value. Without knowing this default behavior, you might mistakenly accept SAML assertions that are issued by an authorized STS

h. Optional: You can add a list of non-root certificate authority (CA) certificates that can be used to check the signature of the SAML token. To add non-root certificates, add a custom property named `X509PATH_n` where *n* is a non-negative integer as the value for the non-root certificates.

i. Optional: You can add a list of certificate revocation lists (CRLs) that can be used to validate the signature of the SAML token. To add CRLs, add a custom property named `CRLPATH_n` where *n* is a non-negative integer as the value for the CRLs.

j. Click **Apply** and **Save**.

18. Optional: You can configure the caller binding to select a SAML token to represent the requester identity. The Web Services Security runtime environment uses the specified JAAS login configuration to acquire the user security name and group membership data from the user registry using the SAML token NameId or NameIdentifier as the user name.

    a. Click **Applications** > **Application types** > **WebSphere enterprise applications** > ***JaxWSServicesSamples*** > **Service provider policy sets and bindings** > **Saml HoK Symmetric Provider sample** > **WS-Security** > **Callers**.

    b. Click **New** to create the caller configuration

    c. Specify a **Name**, such as `caller`.

    d. Enter a value for the **Caller identity local part**. For example, `http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0`, which is the local part of the CustomToken element in the attached WS-Security policy.

    e. Click **Apply** and **Save**.

19. Restart the web services provider application so that the policy set attachment modifications can take effect.

Results

When you have completed the procedure, the JaxWSServicesSamples web services application is ready to use the SAML20 HoK Symmetric default policy set, the Saml HoK Symmetric Client sample, and the Saml HoK Symmetric Provider sample general bindings.

SAMLIssuerConfig.properties file:

When creating a new SAML token, you can specify configuration properties to control how the token is configured. The configuration properties are stored in a properties file containing name/value pairs. The properties describe provider-side information such as the issuer location, and the keystore and truststore file paths.

Starting with WebSphere Application Server version 8, you can also use the administrative console or the `setSAMLIssuerConfigInBinding` command task to specify a self-issued SAML token's configuration as custom properties in the requester's outbound configuration in the general bindings or in the application-specific bindings. You can also specify a self-issued SAML token's configuration as custom properties of `com.ibm.websphere.wssecurity.wssapi.WSSGenerationContext` objects when programming to Web Services Security (WSS) Application Programming interfaces (APIs). Migrate self-issued SAML token configuration data from the `SAMLIssuerConfig.properties` file to the bindings. Refer to the "Managing self-issue SAML token configuration using wsadmin commands" section for additional information.

The `SAMLIssuerConfig.properties` file usage is deprecated in WebSphere Application Server version 8. Do not specify a `SAMLIssuerConfig.properties` file using a Java System property. The `com.ibm.websphere.wssecurity.wssapi.token.SAMLTokenFactory.newDefaultProviderConfig()` method returns a `com.ibm.wsspi.wssecurity.saml.config.ProviderConfig` object with empty contents when no `SAMLIssuerConfig.properties` file is specified, which is the recommended programming style. Use `ProviderConfig` setter methods to populate its contents.

File Location

A single configuration file, SAMLIssuerConfig.properties, containing the provider-side properties is created and stored on each server. On a WebSphere server, the file is located in the server-level repository, or in the cell-level repository. In an environment that is not based on WebSphere, the file location is defined by a Java system property. The name of this property is `com.ibm.webservices.wssecurity.platform.SAMLIssuerConfigDataPath`.

For example, the location of the file at the server level on a WebSphere server is:

app_server_root`/profiles/$PROFILE/config/cells/$CELLNAME/nodes/$NODENAME/servers/$SERVERNAME/SAMLIssuerConfig.properties`

The location of the file at the cell level on a WebSphere server is:

app_server_root`/profiles/$PROFILE/config/cells/$CELLNAME/sts/SAMLIssuerConfig.properties`

SAML token properties

The following table describes the provider configuration properties.

Table 289. Properties to configure provider information for a new SAML token. Use these properties to control how the token is created.

Property name	Sample property value	Property description
IssuerURI	http://www.websphere.ibm.com/SAML/SelfIssuer	The URI of the issuer.
TimeToLiveMilliseconds	3600000	Amount of time before expiration of the token.
KeyStoreRef	*MyKeyStoreRef*	A reference to a managed keystore from security.xml.
KeyStorePath	*app_server_root*/etc/ws-security/samples/dsig-receiver.ks	The location of the keystore file. **Note:** You must modify this value from the default value to match the path location for your system.
KeyStoreType	JKS	The keystore type.
KeyStorePassword	*password*	The password of the keystore file (the password must be XOR encoded). For more information, read about encoding passwords in files.

Table 289. Properties to configure provider information for a new SAML token (continued). Use these properties to control how the token is created.

Property name	Sample property value	Property description
KeyAlias	*soapprovider*	The alias of the key as defined in the keystore file.
KeyName	CN=SOAPProvider, OU=TRL, O=IBM, ST=Kanagawa, C=JP	The name of the key as defined in the keystore file.
KeyPassword	*password*	The password of the private key as defined in the keystore file (the password must be XOR encoded).
TrustStoreRef	*MyTrustStoreRef*	A reference to a managed keystore from security.xml.
TrustStorePath	*app_server_root*/etc/ws-security/samples/dsig-receiver.ks	The location of the truststore file. **Note:** You must modify this value from the default value to match the path location for your system.
TrustStoreType	JKS	The truststore type.
TrustStorePassword	*password*	The password of the truststore file.
AttributeProvider	*com.mycompany.SAML.AttributeProviderImpl*	Implementation class of attribute provider.
NameIDProvider	*com.mycompany.SAML.NameIDProviderImpl*	Implementation class of name ID provider.

Example

See the following example of a SAML token configuration properties file:

```
IssuerURI=http://www.websphere.ibm.com/SAML/SelfIssuer
TimeToLiveMilliseconds=3600000
KeyStorePath=${USER_INSTALL_ROOT}/etc/ws-security/samples/dsig-receiver.ks
KeyStoreType=JKS
KeyStorePassword={xor}LDotKTot
KeyAlias=soapprovider
KeyName=CN=SOAPProvider, OU=TRL, O=IBM, ST=Kanagawa, C=JP
KeyPassword={xor}LDotKTot
TrustStorePath=${USER_INSTALL_ROOT}/etc/ws-security/samples/dsig-receiver.ks
TrustStoreType=JKS
TrustStorePassword={xor}LDotKTot
```

Configuring client and provider bindings for the SAML sender-vouches token:

Configure the client and provider policy set attachments and bindings for the SAML sender-vouches token, which includes the sender-vouches confirmation method. The sender-vouches confirmation method is used when a server needs to propagate the client identity or behavior of the client.

Before you begin

This function is enabled in WebSphere Application Server Version 7.0.0.9 and later releases. To use the function, you must first install WebSphere Application Server Version 7.0.0.9, which includes SAML sender-vouches support. After installing Version 7.0.0.9, you must create one or more new server profiles, or add SAML configuration settings to an existing profile. For example, in a WebSphere Application Server, Network Deployment environment, there are multiple profiles. Read about setting up the SAML configuration for more information. The sender-vouches token must be protected using either message-level security or HTTPS transport. Therefore, you must determine which type of security you want to use.

About this task

A SAML sender-vouches token is a SAML token that uses the sender-vouches subject confirmation method. A SOAP message sender is required to protect the integrity of SOAP messages and SAML tokens so that a receiver can verify that the message contents and SAML tokens were not modified by unauthorized parties.

WebSphere Application Server with SAML provides numerous default SAML token application policy sets and several general client and provider binding samples. The policy set for the SAML sender-vouches token is similar to the SAML bearer token policy set. The procedure shows how to create a sender-vouches policy set based on the attached SAML bearer token policy set. Before you can configure the client and provider bindings for the SAML sender-vouches token, you must attach SAML bearer token client and provider bindings to the JAX-WS application. For more information about the bearer policy sets, read about configuring client and provider bindings for the SAML bearer token.

You must use application-specific custom bindings instead of general bindings for sender-vouches. Therefore, if you configure sender-vouches policy sets and bindings from attached bearer token policy sets and bindings, you must ensure that the assigned bindings are application-specific bindings.

The procedure for creating the sender-vouches policy set begins with attaching the Web services bearer token policy sets.

Procedure

Complete the associated steps to configure the selected protection method. Follow the first set of steps to protect messages using message-level security, or follow the second set of steps to protect messages using HTTPS transport.

- To protect messages using message-level security, attach the SAML20 Bearer WSSecurity default policy set and configure the associated application-specific bindings.
  1. Attach the SAML20 Bearer WSSecurity default policy set. Refer to steps 1 and 2 in the topic, Configuring client and provider bindings for the SAML bearer token.
  2. Create and configure application-specific bindings for the client and the service provider. Refer to steps 3 to 9 in the topic Configuring client and provider bindings for the SAML bearer token. For the binding names, enter names that include sender-vouches, such as SAML20SenderVouches_Client and SAML20SenderVouches_Service.
  3. The sender of SOAP messages (attesting entity) satisfies the sender-vouches subject confirmation requirement by including a <ds:Signature> element in the corresponding <wsse:Security> header. The initiator key is used to sign the relevant message content and assertions. Modify the sender-vouches bindings to satisfy the vouching requirement.
     a. Refer to the topic, Signing SAML tokens at the message level for information about modifying the sender-vouches bindings.
     b. Edit both the **Client policy set and bindings** and the **Service provider policy sets and bindings**. Click **WS-Security** > **Authentication and protection**.
     c. Click the name of the authentication token that is configured in the attached SAML bearer policy set; for example, **request:SAMLToken20Bearer**.
     d. Click **Callback handler**.
     e. Under Custom Properties, select **confirmationMethod**.
     f. Click **Edit**.
     g. Change the value of the confirmationMethod property to sender-vouches.
     h. Verify that the value of the `keyType` property value is `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer` or the `bearer` alias. The `wstrustClientWSTNamespace` property determines how the `bearer` alias is interpreted. In this case it is assumed that it is set to the WS-Trust 1.3 namespace. If it had a value of WS-Trust 1.2, the `bearer` alias carries no meaning, as it is not defined for WS-Trust 1.2.
- To protect messages using HTTPS transport, attach the SAML20 Bearer WSHHTPS default policy set and configure the associated application-specific bindings.
  1. Attach the SAML20 Bearer WSHHTPS default policy set. Refer to steps 1 and 2 in the topic, Configuring policy sets and bindings to communicate with the Security Token Service (STS).

2. Create and configure application-specific bindings for the client and the service provider. Refer to steps 3 to 5 in the topic, Configuring policy sets and bindings to communicate with STS. For the binding names, enter names that include sender-vouches, such as sslSamlSenderVouches_Client and sslSamlSenderVouches_Service.
3. Set the required SAML SubjectConfirmation method to the sender-vouches method.
   a. Edit both the **Client policy set and bindings** and the **Service provider policy sets and bindings**. Click **WS-Security > Authentication and protection**.
   b. Click the name of the authentication token configured in the attached SAML bearer policy set. For example, **request:SAMLToken20Bearer**.
   c. Click **Callback handler**.
   d. Under Custom Properties, select **confirmationMethod**.
   e. Click **Edit**.
   f. Change the value of the confirmationMethod property to sender-vouches.
   g. Verify that the value of the `keyType` property value is `http://docs.oasis-open.org/ws-sx/ws-trust/200512/Bearer` or the `bearer` alias. The `wstrustClientWSTNamespace` property determines how the `bearer` alias is interpreted. In this case it is assumed that it is set to the WS-Trust 1.3 namespace. If it had a value of WS-Trust 1.2, the `bearer` alias carries no meaning, as it is not defined for WS-Trust 1.2.
4. Use X.509 client certificate authentication over SSL to satisfy the sender-vouches subject confirmation requirement. The configuration steps vary on the SOAP message receiver side depending on whether SSL connections end at the internal HTTP server or at an external HTTP server. An external HTTP server might be a web server, a reverse proxy security server, a proxy server, and so on.

   When client certificate authentication is required, the internal HTTP server accepts a client SSL connection request under one of the following conditions:
   - The X.509 certificate for the client exists in the trust store.
   - The X.509 certificate is issued by a trusted entity or party, which means that the X.509 certificate for the issuer exists in the trust store.
   - If a web services SSL connection ends at an internal HTTP server, complete the following steps:
     a. Click **Security > SSL certificate and key management > Manage endpoint security configurations > *node_name* > SSL configurations**.
     b. Select the resources used in the SSL transport bindings and in the secure transport chain.
     c. Under Additional Properties, click **Quality of protection (QoP) settings**.
     d. Under General Properties, select **Required** from the Client Authentication menu list.
     e. Click **Apply**.
   - If a web services client SSL connection ends at an external HTTP server, complete the following steps:
     a. Regenerate the plug-in. Click **Servers > Web Servers**.
     b. Select the web server, then click **Generate Plug-in**.
     c. Update the HTTP server with the generated plug-in.
     d. Restart the HTTP server.
     e. Enable client certificate authentication from the HTTPS client to your web server. For more information, see the documentation about Secure Sockets Layer (SSL) client certificate authentication.

        The external web server is configured to propagate web services client SSL context data to the application server. Protect the connection between the external web server and the application server to prevent man-in-the-middle attacks.

Managing self-issue SAML token configuration using wsadmin commands:

The SAMLIssuerConfig.properties file usage is deprecated in WebSphere Application Server Version 8. You can use the listSAMLIssuerConfig and updateSAMLIssuerConfig wsadmin command tasks to read and modify the SAMLIssuerConfig.properties cell level and server level configuration files. Starting with WebSphere Application Server Version 8, you should use the administrative console or the setSAMLIssuerConfigInBinding command task to specify a self-issued SAML token's configuration as custom properties in the requester's outbound configuration in the general bindings or in the application-specific bindings. Do not use server level and cell level SAMLIssuerConfig.properties file.

Before you begin

The product provides an alternate way to specify a self-issued SAML token configuration in policy set bindings. Migrate self-issued SAML token configuration data from the SAMLIssuerConfig.properties file to the bindings. Specifying configuration data for creating self-issued SAML tokens in general bindings or application-specific bindings provides management flexibility to specify the configuration at a finer grained scope, in addition to the cell level and the server level. For example you can configure a specific SAML token issuer for a particular web service application, for an arbitrary group of applications, or for a web service application in a security domain.

Note: Self-issued SAML token configuration data that is defined in the bindings takes precedence over data that is defined in the server level or the cell level SAMLIssuerConfig.properties file, in that order. When a self-issued SAML token configuration data is defined in an attached policy set bindings, the Web services security runtime environment will neglect the SAMLIssuerConfig.properties files, both at the server level and at the cell level. So it is important that when you migrate from the SAMLIssuerConfig.properties file to the bindings, you must migrate all the required properties.

About this task

Note: Two command tasks are available to manage the SAMLIssuerConfig.properties file-based SAML issuer configuration. This file can be located at the cell level and the server level. These two tasks are:

- listSAMLIssuerConfig
- updateSAMLIssuerConfig

Procedure

1. Run the wsadmin command task in the interactive mode. The following Jython script illustrates how to run the wsadmin command task in the interactive mode.

```
AdminTask.listSAMLIssuerConfig('[-interactive]')
```

   To select the server level SAML issuer configuration, the *serverName* and *nodeName* parameters are required. If these parameters are missing, then the command task lists the cell level SAML issuer configuration.

2. Use the listSAMLIssuerConfig command task to display the server level SAML issuer configuration.

```
AdminTask.listSAMLIssuerConfig('[-nodeName Node01 -serverName server1]')
```

   You need the “monitor” or above administrative role privilege to execute the listSAMLIssuerConfig command.

3. Use the updateSAMLIssuerConfig command task to update the server level or cell level SAML issuer configuration.

```
AdminTask.updateSAMLIssuerConfig('[-IssuerURI My_Issuer
                                  -TimeToLiveMilliseconds 3600000
                                  -KeyStoreRef "name=myKeyStore managementScope=(cell):Node01Cell:(node):Node01"
                                  -KeyAlias samlissuer
                                  -KeyName "CN=SAMLIssuer, O=Acme, C=US" -KeyPassword *****
                                  -TrustStoreRef "name=myKeyStore managementScope=(cell):Node01Cell:(node):Node01 "]')
```

   If the *serverName* and *nodeName* parameters are not specified, then the task updates the cell level SAML issuer configuration.

You need the “administrator” administrative role privilege to execute the `updateSAMLIssuerConfig` command.

Results

You have created command scripts to automate the process of updating the cell level or the server level `SAMLIssuerConfig.properties` files, or you have created self-issued SAML token configuration data as custom properties in the requester's outbound configuration in the general bindings or in the application-specific bindings.

Example

The following example illustrates how to add or modify self-issued SAML token configuration data in the application-specific bindings:

```
AdminTask.setSAMLIssuerConfigInBinding('[-bindingName SAMLTestAppClientBinding
-bindingLocation [ [application JaxWSServicesSamples] [attachmentId 1904] ]
-com.ibm.wsspi.wssecurity.saml.config.issuer.IssuerURI My_Issuer
-com.ibm.wsspi.wssecurity.saml.config.issuer.TimeToLiveMilliseconds 3600000
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyStoreRef "name=myKeyStore managementScope=(cell):Node01Cell:(node):Node01 "
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyAlias samlissuer
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyName "CN=SAMLIssuer, O=Acme,C=US"
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyPassword *****
-com.ibm.wsspi.wssecurity.saml.config.issuer.TrustStoreRef "name=myKeyStore managementScope=(cell):Node01Cell:(node):Node01 "]')
```

The following example illustrates how to modify the general bindings:

```
AdminTask.setSAMLIssuerConfigInBinding('[-bindingName "Saml Bearer Client sample"
-bindingScope domain -bindingLocation  -domainName global
-com.ibm.wsspi.wssecurity.saml.config.issuer.IssuerURI My_Issuer
-com.ibm.wsspi.wssecurity.saml.config.issuer.TimeToLiveMilliseconds 3600000
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyStorePath "profile_root/etc/ws-security/saml/saml-issuer.jceks
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyStoreType jceks
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyStorePassword *****
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyAlias samlissuer
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyName "CN=SAMLIssuer, O=Acme, C=US"
-com.ibm.wsspi.wssecurity.saml.config.issuer.KeyPassword *****
-com.ibm.wsspi.wssecurity.saml.config.issuer.TrustStorePath "profile_root/profiles/<server_name>/etc/ws-security/saml/saml-issuer.jceks
-com.ibm.wsspi.wssecurity.saml.config.issuer.TrustStoreType jceks
-com.ibm.wsspi.wssecurity.saml.config.issuer.TrustStorePassword *****]')
```

When specifying the application bindings, **bindingLocation** is a required parameter and can be supplied as a properties object. The property names are `application` and `attachmentId`. When specifying the general bindings, **bindingLocation**, which can be null or have empty properties, is required. Additionally, **bindingScope** is required if the scope is not global. Use the **bindingName** parameter to identify the binding location. For more information about **bindingLocation**, **bindingScope**, and **domainName**, refer to the setBinding or getBinding command tasks documentation.

To remove SAML issuer configuration custom properties from the bindings, use the administrative console or the setBinding command task.

Configuring default Web Services Security bindings

WebSphere Application Server provides support for a set of default Web Services Security bindings for applications. A set of bindings is a named object that is associated with a specific policy set and service resource attached to the policy set.

About this task

Bindings contain environment and platform specific information, such as the following types of information:

- Keys used for signature and encryption
- Keystore information
- Authentication information
- Persistent information

In WebSphere Application Server Version 7.0 and later, there are two types of bindings, application specific bindings and general bindings. Typically, bindings are specific to the application or the platform, and they are not shared.

General bindings can be configured to be used across a range of policy sets and can be reused across applications and for trust service attachments. Though general bindings are highly reusable, they are not able to provide configuration for advanced policy requirements, such as multiple signatures. There are two types of general bindings: general provider policy set bindings and general client policy set bindings. The general bindings that are shipped with WebSphere Application Server are initially set as the default bindings, but you can choose a different binding as the default, or change the level of binding that should be used as the default, for example, from cell level binding to server level binding. Default bindings are used when no application specific binding or trust service binding has been assigned to a policy set attachment. For more information, see the topic General JAX-WS default bindings for Web Services Security. For a description of the general sample bindings that are included with WebSphere Application Server, and used with the JAX-WS programming model, read the topic General sample bindings for JAX-WS applications.

To create general bindings:

Procedure

1. Log in to the administrative console and navigate to the general provider policy set and bindings panel, or the general client policy set and bindings panel
   - Click **Services > Policy sets > General provider policy set bindings**.
   - Click **Services > Policy sets > General client policy set bindings**.
2. Click **New**.

Results

Policy set bindings contain platform-specific information, like keystore, authentication information or persistent information, required by a policy set attachment. Each policy set attachment to a service provider or service client must have exactly one binding. When you create a policy set attachment, the general default bindings are used initially. When general bindings are used in association with a policy set attachment, the cell-level general bindings are applied at run time. If application server level bindings exist, the server-level general bindings override the cell-level definition. General bindings specify configuration for both service client and service provider attachments and the general bindings are not tailored to a specific policy set or application. When you define server-level general bindings, the binding begins in a completely unconfigured state. You must add the policy, and then fully configure the bindings for each added policy.

An application specific binding is a named binding that you create. Application specific bindings enable you to provide platform-specific configuration information for specific policy set attachments. When you create an application specific binding, the available binding configuration options are tailored to the definitions in the attached policy set. You can reuse application specific bindings for multiple service resources within an application. For example, if you create a trust service specific binding, that binding can be reused only for trust service attachments. When you create an application specific binding for a policy set attachment, the binding begins in a completely unconfigured state. For each policy, such as WS-Security or HTTP Transport, where you want to override the general binding, you must add the policy, and then fully configure the bindings for each added policy.

Important: Only use the sample default bindings in a testing environment. Do not use sample default bindings in a production environment. Default bindings contain sample key files that must be customized before use in a production environment.

See the topic Defining and managing service client or provider bindings for more information about bindings.

General JAX-WS default bindings for Web Services Security

General bindings are used as the default bindings at the cell level or server level, or for multiple domains, at the domain level. The general bindings that are included with WebSphere Application Server are initially set as the default bindings. However, you can choose a different binding as the default, or change the level of binding that is used as the default, for example, from cell-level binding to server-level binding.

Policy set bindings contain platform-specific information, such as keystore, authentication information or persistent information, required by a policy set attachment. In WebSphere Application Server Version 7.0 and later, there are two types of bindings: application-specific bindings, and general bindings. Both types of bindings are supported for WS-Security policy sets. General bindings can be used as default bindings, and can also be shared across multiple applications and for trust service attachments. There are two types of general bindings: one for service providers and one for service clients. You can define multiple general bindings for the provider and also for the client. However, only one general provider binding and one general client binding can be designated as the default.

Default bindings are used when no application-specific binding or trust service binding has been assigned to a policy set attachment. You can choose the general provider and general client bindings, which are used as the default bindings for the cell. These are the global security settings. Likewise, you can choose the general provider and general client bindings, which are used as the default bindings for a server. For specific information about selecting bindings, see the topic Defining and managing policy set bindings.

In an environment with multiple security domains, you can also choose the general provider and general client bindings, which are used as the default bindings for a domain. If you do not choose a binding to be the default for a server, the default bindings for the domain in which the server resides are used. If you do not choose a binding to be the default for a domain, the default bindings for the cell (global security) are used. You must choose default provider and default client bindings for the cell.

The general bindings that are included with WebSphere Application Server are initially set as the cell default bindings. You cannot delete a binding that has been selected as the default binding for server, a domain, or the cell. Before you delete a binding that is selected as the default, you must select a different default binding, or specify that the defaults for the cell (global security) should be used.

The following default bindings are shipped with the product:

- Provider sample
- Client sample
- Version 6.1 default policy set bindings

  The Version 6.1 bindings are used only if a WebSphere Application Server Version 6.1 Feature Pack for Web Services application is installed within the WebSphere Application Server Version 7.0 and later environment. For more information on these bindings, see the topic Version 6.1 default policy set bindings.

Important: Do not use the provider and client sample bindings that are included with WebSphere Application Server in their current state in a production environment. You must modify these bindings to meet your security needs before using them in a production environment by making a copy of the bindings and then modifying the copy. For example, change the key and keystore settings to ensure security, and modify the binding settings to match your environment.

For a detailed description of the general sample bindings, see the topic General sample bindings for JAX-WS applications.

To define and manage general bindings, in the administrative console click **Services > Policy sets > General provider policy set bindings** or **Services > Policy sets > General client policy set bindings**. To manage bindings for the cell or the domain, click **Services > Policy sets > Default policy set**

bindings. The general service provider and client bindings have independent settings that you can customize to meet the needs of your environment. To learn more about general bindings, read the topic Defining and managing policy set bindings.

In addition to choosing default bindings for the cell (global security), you can also choose the general provider and general client bindings that you want to use as the default bindings for a server. When are using the JAX-WS programming model and want to specify the server default bindings, log on to the administrative console and click **Servers > Server Types > WebSphere application servers > *server_name***. In the Security section of the console page, click **Default policy set bindings**.

Administering message-level security for JAX-RPC web services

The Java™ API for XML-based RPC (JAX-RPC) specification enables you to develop SOAP-based interoperable and portable web services and web service clients. JAX-RPC simplifies development of web services by shielding you from the underlying complexity of SOAP communication, and enables clients to access a web service as if the web service was a local object mapped into the client's address space.

Securing messages using JAX-RPC at the request and response generators

You can secure messages with tokens and encryption to protect message integrity, authenticity, and confidentiality.

About this task

To secure messages, you can:

- Configure generator signing to protect message integrity
- Configure encryption to protect message confidentiality at the server level and at the application level
- Configure tokens to protect message authenticity at the server level and at the application level

Procedure

- To configure generator signing to protect message integrity, see the steps outlined in “Configuring generator signing using JAX-RPC to protect message integrity” on page 3294.
- To configure encryption to protect message confidentiality at the application level, see the steps outlined in “Configuring encryption using JAX-RPC to protect message confidentiality at the application level” on page 3355.
- To configure encryption to protect message confidentiality at the server level, see the steps outlined in “Configuring encryption using JAX-RPC to protect message confidentiality at the server or cell level” on page 3378.
- To configure tokens to protect message authenticity at the application level, see the steps outlined in “Configuring token generators using JAX-RPC to protect message authenticity at the application level” on page 3327.
- To configure tokens to protect message authenticity at the server level, see the steps outlined in “Configuring token generators using JAX-RPC to protect message authenticity at the server level” on page 3382.

Results

By completing the steps in the previous tasks, you have secured messages using tokens and encryption to protect message integrity, authenticity, and confidentiality.

Securing messages using JAX-RPC at the request and response consumers

You can secure messages at the request and response consumer level to protect message confidentiality and security.

About this task

To secure messages, you can:

- Configure signing to protect message confidentiality
- Configure encryption to protect message confidentiality at the server level and at the application level
- Configure tokens to protect message authenticity at the server level and at the application level

Procedure

- To configure consumer signing to protect message confidentiality, see the steps outlined in “Configuring consumer signing using JAX-RPC to protect message integrity” on page 3309
- To configure encryption to protect message confidentiality at the application level, see the steps outlined in “Configuring encryption to protect message confidentiality at the application level” on page 3367.
- To configure encryption to protect message confidentiality at the server level, see the steps outlined in “Configuring encryption to protect message confidentiality at the server level” on page 3380.
- To configure tokens at the application level to protect message authenticity, see the steps outlined in “Configuring token consumers using JAX-RPC to protect message authenticity at the application level” on page 3344.
- To configure tokens at the server level to protect message authenticity, see the steps outlined in “Configuring token consumers using JAX-RPC to protect message authenticity at the server level” on page 3393.

Results

By completing the steps in the previous tasks, you have secured messages at the request and response consumer level.

Configuring message-level security for JAX-RPC at the application level

Modify the application-level configurations in the administrative console.

Configuring generator signing using JAX-RPC to protect message integrity:

You can configure the generator key and signing information at the server and application level to protect message integrity.

About this task

To protect message integrity, you can:

- Configure the signing information for the client-side request generator and the server-side response generator bindings at the server level and at the application level
- Configure the key information for the request generator (client side) and the response generator (server side) bindings on the server level and at the application level

Procedure

- To configure the signing information for the client-side request generator and the server-side response generator bindings at the server , see the steps outlined in “Configuring the signing information using JAX-RPC for the generator binding on the server level” on page 3369.
- To configure the signing information for the client-side request generator and the server-side response generator bindings at the application level, see the steps outlined in “Configuring the signing information using JAX-RPC for the generator binding on the application level” on page 3295
- To configure the key information for the request generator (client side) and the response generator (server side) bindings at the application level, see the steps outlined in “Configuring the key information using JAX-RPC for the generator binding on the application level” on page 3314.

- To configure the key information for the generator binding on the server level, see the steps outlined in "Configuring the key information for the generator binding using JAX-RPC on the server level" on page 3375.

Results

By completing the steps in these tasks, you have configured generator signing to protect the integrity of messages.

Configuring the signing information using JAX-RPC for the generator binding on the application level:

You can configure the signing information for the client-side request generator and the server-side response generator bindings at the application level.

Before you begin

Note: For WebSphere Application Server version 6.x or earlier only, in the server-side extensions file (`ibm-webservices-ext.xmi`) and the client-side deployment descriptor extensions file (`ibm-webservicesclient-ext.xmi`), you must specify which parts of the message are signed. Also, you must configure the key information that is referenced by the key information references on the signing information panel within the administrative console.

About this task

This task explains the required steps to configure the signing information for the client-side request generator and the server-side response generator bindings at the application level. WebSphere Application Server uses the signing information for the default generator to sign parts of the message including the body, time stamp, and user name token. The Application Server provides default values for bindings. However, an administrator must modify the defaults for a production environment. Complete the following steps to configure the signing information for the generator sections of the bindings files on the application level:

Procedure

1. Locate the signing information configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties, you can access the signing information for the request generator and the response generator bindings.
      - For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
      - For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
   d. Under Required properties, click **Signing information**.
   e. Click **New** to create a signing information configuration, select the box next to the configuration and click **Delete** to delete an existing configuration, or click the name of an existing signing information configuration to edit its settings. If you are creating a new configuration, enter a name in the Signing information name field. For example, you might specify `gen_signinfo`.
2. Select a signature method algorithm from the Signature method field. The algorithm that is specified for the generator, which is either the request generator or the response generator configuration, must match the algorithm that is specified for the consumer, which is either the request consumer or response consumer configuration. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2000/09/xmldsig#rsa-sha1

- http://www.w3.org/2000/09/xmldsig#hmac-sha1
- http://www.w3.org/2000/09/xmldsig#dsa-sha1

Restriction: Do not use this algorithm if you want the configured application to be compliant with the Basic Security Profile (BSP).

Any ds:SignatureMethod/@Algorithm element in a SIGNATURE based on a symmetric key must have a value of http://www.w3.org/2000/09/xmldsig#rsa-sha1 or http://www.w3.org/2000/09/xmldsig#hmac-sha1.

3. Select a canonicalization method from the **Canonicalization method** field. The canonicalization algorithm that you specify for the generator must match the algorithm for the consumer. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/10/xml-exc-c14n#
   - http://www.w3.org/2001/10/xml-exc-c14n#WithComments
   - http://www.w3.org/TR/2001/REC-xml-c14n-20010315
   - http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
4. Select a key information signature type from the Key information signature type field. WebSphere Application Server supports the following signature types:

   None Specifies that the <KeyInfo> element is not signed.

   Keyinfo
   Specifies that the entire <KeyInfo> element is signed.

   Keyinfochildelements
   Specifies that the child elements of the <KeyInfo> element are signed.

   The key information signature type for the generator must match the signature type for the consumer. You might encounter the following situations:
   - If you do not specify one of the previous signature types, WebSphere Application Server uses `keyinfo`, by default.
   - If you select Keyinfo or Keyinfochildelements and you select http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform as the transform algorithm in a subsequent step, WebSphere Application Server also signs the referenced token.
5. Select a signing key information reference from the Signing key information field. This selection is a reference to the signing key that the Application Server uses to generate digital signatures.
6. Click **OK** and **Save** to save the configuration.
7. Click the name of the new signing information configuration. This configuration is the one that you specified in a previous step.
8. Specify the part reference, digest algorithm, and transform algorithm. The part reference specifies which parts of the message to digitally sign.
   a. Under Additional properties, click **Part references** > **New** to create a new part reference, click **Part references** > **Delete** to delete an existing part reference, or click a part name to edit an existing part reference.
   b. Specify a unique part name for this part reference. For example, you might specify `reqint`.
   c. Select a part reference from the Part reference field.

      The part reference refers to the message part that is digitally signed. The part attribute refers to the name of the <Integrity> element in the deployment descriptor when the <PartReference> element is specified for the signature. You can specify multiple <PartReference> elements within the <SigningInfo> element. The <PartReference> element has two child elements when it is specified for the signature: <DigestTransform> and <Transform>.
   d. Select a digest method algorithm from the menu. The digest method algorithm specified within the <DigestMethod> element is used in the <SigningInfo> element.

      WebSphere Application Server supports the following algorithms:

- http://www.w3.org/2000/09/xmldsig#sha1
- http://www.w3.org/2001/04/xmlenc#sha256
- http://www.w3.org/2001/04/xmlenc#sha512

e. Click **OK** to save the configuration.

f. Click the name of the new part reference configuration. This configuration is the one that you specified in a previous step.

g. Under Additional Properties, click **Transforms** > **New** to create a new transform, click **Transforms** > **Delete** to delete a transform, or click a transform name to edit an existing transform. If you create a new transform configuration, specify a unique name. For example, you might specify `reqint_body_transform1`.

h. Select a transform algorithm from the menu. The transform algorithm is that is specified within the <Transform> element and specifies the transform algorithm for the signature. WebSphere Application Server supports the following algorithms:

- http://www.w3.org/2001/10/xml-exc-c14n#
- http://www.w3.org/TR/1999/REC-xpath-19991116

  Restriction: Do not use this transform algorithm if you want your configured application to be compliant with the Basic Security Profile (BSP). Instead use http://www.w3.org/2002/06/xmldsig-filter2 to ensure compliance.
- http://www.w3.org/2002/06/xmldsig-filter2
- http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform
- http://www.w3.org/2002/07/decrypt#XML
- http://www.w3.org/2000/09/xmldsig#enveloped-signature

The transform algorithm that you select for the generator must match the transform algorithm that you select for the consumer.

Important: If both of the following conditions are true, WebSphere Application Server signs the referenced token:

- You previously selected the Keyinfo or the Keyinfochildelements option from the Key information signature type field on the signing information panel.
- You select http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform as the transform algorithm.

9. Click **Apply**.
10. Optional: Determine whether to disable the Inclusive namespace prefix list. The Exclusive XML Canonicalization Version 1.0 specification recommends that you include all of the namespace declarations that correspond to the namespace prefix in the canonicalization form. For security reasons, WebSphere Application Server, by default, includes the prefix in the digital signature for Web Services Security. However, some implementations of Web Services Security cannot handle this prefix list. WebSphere Application Server can handle digitally signed messages that either contain or do not contain the prefix list. If you experience a signature validation failure when a signed Simple Object Access Protocol (SOAP) message is sent and you are using another vendor in your environment, check with your service provider for a possible fix to their implementation before you disable this property. To disable this property, complete the following steps:

a. Under Additional properties, click **Properties** > **New**.

b. In the Property name field, enter the `com.ibm.wsspi.wssecurity.dsig.inclusiveNamespaces` property.

c. In the Property value field, enter the `false` value.

d. Click **OK**.

You can set this property for both the request generator and the response generator configurations.

11. Click **Save** at the top of the panel to save your configuration.

Results

After completing these steps, the signing information is configured for the generator on the application level.

What to do next

You must specify a similar signing information configuration for the consumer.

Signing information collection:

Use this page to view a list of signing parameters. Signing information is used to sign and validate parts of a message including the body, time stamp, and user name token. You can also use these parameters for X.509 validation when the authentication method is IDAssertion and the ID type is X509Certificate in the server-level configuration. In such cases, you must fill in the certificate path fields only.

To view this administrative console page on the server level for signing information, complete the following steps:

1. Click **Servers > Server Types > WebSphere application Servers > *server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Signing information**.
4. Click **New** to create a signing parameter. Click **Delete** to delete a signing parameter.

To view this administrative console page on the application level for signing information, complete the following steps:

1. Click **Applications > Application Types > WebSphere enterprise applications > *application_name***.
2. Under Modules, click **Manage modules > *URI_name***.
3. Under Web Services Security Properties, you can access the signing information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
4. Under Required properties, click **Signing information**.
5. Under Additional properties, you can use this panel to configure the following bindings:
   - For the Request receiver binding, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**.
   - For the Response receiver binding, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**.
6. Under Additional properties, click **Signing information**.
7. Click **New** to create a signing parameter. Click **Delete** to delete a signing parameter.

Signing information name:

Specifies the unique name that is assigned to the signing configuration.

Signature method:

Specifies the signature method algorithm that is chosen for the signing configuration.

Canonicalization method:

Specifies the canonicalization method algorithm that is chosen for the signing configuration.

Signing information configuration settings:

Use this page to configure new signing parameters.

The specifications that are listed on this page for the signature method, digest method, and canonicalization method are located in the World Wide Web Consortium (W3C) document entitled, *XML Signature Syntax and Specification: W3C Recommendation 12 Feb 2002*.

To view this administrative console page on the server level for signing information, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Signing information**.
4. Click **New** to create a signing parameter or click the name of an existing configuration to modify its settings.

To view this administrative console page on the application level for signing information, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access the signing information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
4. Under Required properties, click **Signing information**.
5. Under Additional properties, you can access the signing information for the following bindings:
   - For the Request receiver binding, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**.

- For the Response receiver binding, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**.

6. Under Additional properties, click **Signing information**.
7. Click **New** to create a signing parameter or click the name of an existing configuration to modify its settings.

Signing information name:

Specifies the name that is assigned to the signing configuration.

Signature method:

Specifies the algorithm Uniform Resource Identifiers (URI) of the signature method.

The following pre-configured algorithms are supported:
- http://www.w3.org/2000/09/xmldsig#rsa-sha1
- http://www.w3.org/2000/09/xmldsig#dsa-sha1

  Do not use this algorithm if you want the configured application to be compliant with the Basic Security Profile (BSP). Any ds:SignatureMethod/@Algorithm element in a signature based on a symmetric key must have a value of http://www.w3.org/2000/09/xmldsig#rsa-sha1 or http://www.w3.org/2000/09/xmldsig#hmac-sha1.
- http://www.w3.org/2000/09/xmldsig#hmac-sha1

For Version 6.0.*x* applications, you can specify additional signature methods on the Algorithm URI panel. To access the Algorithm URI panel, complete the following steps:
1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Algorithm mappings** > ***algorithm_factory_engine_class_name*** > **Algorithm URI** > **New**.

When you specify the Algorithm URI, you also must specify an algorithm type. To have the algorithm display as a selection in the Signature method field on the Signing information panel, you must select **Signature** as the algorithm type.

This field is available for Version 6.*x* and later applications.

Digest method:

Specifies the algorithm URI of the digest method.

The http://www.w3.org/2000/09/xmldsig#sha1 algorithm is supported.

Canonicalization method:

Specifies the algorithm URI of the canonicalization method.

The following pre-configured algorithms are supported:
- http://www.w3.org/2001/10/xml-exc-c14n#
- http://www.w3.org/2001/10/xml-exc-c14n#WithComments
- http://www.w3.org/TR/2001/REC-xml-c14n-20010315
- http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments

This field is for Version 6.*x* and later applications.

Key information signature type:

Specifies how to sign a KeyInfo element if `dsigkey` or `enckey` is specified for the signing part in the deployment descriptor.

This product supports the following keywords:

keyinfo (default)
Specifies that the entire KeyInfo element is signed.

keyinfochildelements
Specifies that the child elements of the KeyInfo element is signed.

If you do not specify a keyword, the application server uses the KeyInfo value, by default.

The Key information signature type field is available for the token consumer binding.

For Version 6.0.*x* applications, this field is also available for the default consumer, request consumer, and response consumer bindings.

Signing key information:

Specifies a reference to the key information that the application server uses to generate the digital signature.

You can specify one signing key only for the default generator binding on the server level. However, you can specify multiple signing keys for the default consumer bindings. The signing keys for the default consumer bindings are specified using the Key Information references link under Additional properties on the Signing information panel.

On the application level, you can specify only one signing key for the request generator and the response generator. You can specify multiple signing keys for the request consumer and response generator. The signing keys for the request consumer and the response consumer are specified using the Key information references link under Additional properties.

You can specify a signing key configuration for the following bindings on the following levels:

Table 290. Signing key binding information. The key is used for digital signature of messages.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under JAX-RPC Default Generator Bindings, click **Key information**.

Table 290. Signing key binding information (continued). The key is used for digital signature of messages.

Binding name	Server level or application level	Path
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under JAX-RPC Default Consumer Bindings, click **Key information**.

Certificate path:

Specifies the settings for the certificate path validation. When you select **Trust any**, this validation is skipped and all incoming certificates are trusted.

The certificate path options are available in token consumer attributes.

Trust anchor

The application server searches for trust anchor configurations on the application and server levels and lists the configurations in this menu.

You can specify trust anchors as an additional property for the response receiver binding and the request receiver binding.

You can specify a trust anchor configuration for the following bindings on the following levels:

Table 291. Trust anchor binding information. The trust anchor is used for signing messages.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Trust anchors > New**.
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Trust anchors** > **New**.
Response receiver	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. 4. Under the Response receiver binding, click **Edit**. 5. Under Additional properties, click **Trust anchors** > **New**.

Table 291. Trust anchor binding information (continued). The trust anchor is used for signing messages.

Binding name	Server level or application level	Path
Request receiver	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. 4. Under the Request receiver binding, click **Edit**. 5. Under Additional properties, click **Trust anchors** > **New**.

For an explanation of the fields on the trust anchor panel, see the help topic Trust anchor configuration settings.

Certificate store

The application server searches for certificate store configurations on the application and server levels and lists the configurations in this menu.

You can specify a certificate store configuration for the following bindings on the following levels:

Table 292. Certificate configurations for bindings. The certificate store is used for signing messages.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Collection certificate store** > **New**.
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Collection certificate store** > **New**.
Response receiver	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. 4. Under the Response receiver binding, click **Edit**. 5. Under Additional properties, click **Collection certificate store** > **New**.
Request receiver	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. 4. Under the Request receiver binding, click **Edit**. 5. Under Additional properties, click **Collection certificate store** > **New**.

For an explanation of the fields on the collection certificate store panel, see the help topic Collection certificate store configuration settings.

Part reference collection:

Use this page to view the message part references for signature and encryption that are defined in the deployment descriptors.

To view this administrative console page on the server level for signing information, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Signing information** > ***signing_information_name***.
4. Under Additional properties, click **Part references**.

To view this administrative console page on the application level for signing information, complete the following steps. Part references are available through the administrative console using Version 6.*x* and later applications only.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access the signing information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sending) binding, click **Edit custom**.
   - For Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
4. Under Required properties, click **Signing information** > ***signing_information_name***.
5. Under Additional properties, click **Part references**.

Part name:

Specifies the name that is assigned to the part reference configuration.

Part reference name:

Specifies the name of the signed part that is defined in the deployment descriptor.

The Part reference name field is specified in the application binding configuration only.

Digest method algorithm:

Specifies the algorithm Uniform Resource Identifier (URI) of the digest method that is used for the signed part that is specified by the part reference.

Part reference configuration settings:

Use this page to specify a reference to the message parts for signature and encryption that are defined in the deployment descriptors.

To view this administrative console page on the server level for signing information, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Signing information** > ***signing_information_name***.
4. Under Additional properties, click **Part references**.
5. Click **New** to create a part reference or click the name of an existing configuration to modify its settings.

To view this administrative console page on the application level for signing information, complete the following steps.

Note: Part references are available through the administrative console using Version 6.*x* applications only.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** .
2. Click **Manage modules** > ***URI_name*** .
3. Under Web Services Security Properties, you can access the signing information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sending) binding, click **Edit custom**.
   - For Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
4. Under Required properties, click **Signing information** > ***signing_information_name***.
5. Under Additional properties, click **Part references**.
6. Click **New** to create a part reference or click the name of an existing configuration to modify its settings.

You must specify a part name and select a part reference before specifying additional properties. Before specifying the digest method properties that are accessible under Additional properties, specify a digest method algorithm on this panel. If you specify `none` and click **Digest method**, an error message is displayed.

Part name:

Specifies the name that is assigned to the part reference configuration.

Part reference name:

Specifies the name of the <integrity> or <requiredIntegrity> element for the signed part of the message or it specifies the name of the <confidentiality> or <requiredConfidentiality> element for the encrypted part of the message in the deployment descriptor.

The part names that are defined in the deployment descriptor are listed as options in this field. This field is displayed for the binding configuration on the application level only.

Digest method algorithm:

Specifies the algorithm Uniform Resource Identifier (URI) of the digest method that is used for the signed part that is specified by the part reference.

This product provides the following predefined algorithm URIs:

- `http://www.w3.org/2000/09/xmldsig#sha1`
- `http://www.w3.org/2001/04/xmlenc#sha256`
- `http://www.w3.org/2001/04/xmlenc#sha512`

If you want to specify a custom algorithm, you must configure the custom algorithm in the Algorithm URI panel before setting the digest method algorithm.

To access the Algorithm URI panel, complete the following steps for the server level:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Algorithm mappings** > ***algorithm_factory_engine_class_name*** > **Algorithm URI** > **New**.

The specified algorithms are listed as options for this field.

When you specify the Algorithm URI, you also must specify an algorithm type. To have the algorithm display as a selection in the Digest method algorithm field on the Part reference panel, you must select **Digest value calculation (Message digest)** as the algorithm type.

Transforms collection:

Use this page to view the transform algorithm that is used for processing the Web Services Security message.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To view this administrative console page for the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Signing information** > ***signing_information_name***.
4. Under Additional properties, click **Part references** > ***part_name***.
5. Under Additional properties, click **Transforms**.

To view this administrative console page for the application level, complete the following steps.

Note: This option is available for Version 6 and later applications only.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.

2. Under Modules, click **Manage Modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access the transforms information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click `Edit custom`.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
   - For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
4. Under Required properties, click **Signing information** > ***signing_information_name***.
5. Under Additional properties, click **Part references** > ***part_name*Transforms** .

Transform name:

Specifies the name that is assigned to the transform algorithm.

Transform algorithm:

Specifies the algorithm Uniform Resource Identifier (URI) of the transform algorithm.

Transforms configuration settings:

Use this page to specify the transform algorithm that is used for processing the Web Services Security message.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To view this administrative console page for the server level, complete the following steps:
1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Signing information** > ***signing_information_name***.
4. Under Additional properties, click **Part references** > ***part_name***.
5. Under Additional properties, click **Transforms**.
6. Click **New** to create a transform configuration or click the name of an existing configuration to modify its settings.

To view this administrative console page for the application level, complete the following steps. This option is available for Version 6.*x* applications only.
1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access the transforms information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.

- For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
- For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
- For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.

4. Under Required properties, click **Signing information** > ***signing_information_name***.
5. Under Additional properties, click **Part references** > ***part_name*** > **Transforms**.
6. Click **New** to create a transform configuration or click the name of an existing configuration to modify its settings.

You must specify a transform name and select a transform algorithm before specifying additional properties.

Transform name:

Specifies the name that is assigned to the transform algorithm.

Transform algorithm:

Specifies the algorithm Uniform Resource Identifier (URI) of the transform algorithm.

This product supports the following algorithms:

http://www.w3.org/2001/10/xml-exc-c14n#
This algorithm specifies the World Wide Web Consortium (W3C) Exclusive Canonicalization recommendation.

http://www.w3.org/TR/1999/REC-xpath-19991116
This algorithm specifies the W3C XML path language recommendation. If you specify this algorithm, you must specify the property name and value by clicking **Properties**, which is displayed under Additional properties. For example, you might specify the following information:

Property
`com.ibm.wsspi.wssecurity.dsig.XPathExpression`

Value `not(ancestor-or-self::*[namespace-uri()='http://www.w3.org/2000/09/xmldsig#' and local-name()='Signature'])`

Note: Do not use this transform algorithm if you want your configured application to be compliant with the Basic Security Profile (BSP). Instead use http://www.w3.org/2002/06/xmldsig-filter2 to ensure compliance.

http://www.w3.org/2002/06/xmldsig-filter2
This algorithm specifies the XML-Signature XPath Filter Version 2.0 proposed recommendation.

When you use this algorithm, you must specify a set of properties. You can use multiple property sets for the XPath Filter Version 2. Therefore, it is recommended that your property names end with the number of the property set, which is denoted by an asterisk in the following examples:

- To specify an XPath expression for the XPath filter2, you might use:

```
name com.ibm.wsspi.wssecurity.dsig.XPath2Expression_*
```

- To specify a filter type for each XPath, you might use:

```
name com.ibm.wsspi.wssecurity.dsig.XPath2Filter_*
```

Following this expression, you can have a value, `[intersect]`, `[subtract]`, or `[union]`.

- To specify the processing order for each XPath, you might use:

```
name com.ibm.wsspi.wssecurity.dsig.XPath2Order_*
```

Following this expression, indicate the processing order of the XPath.

The following is a list of complete examples:

```
com.ibm.wsspi.wssecurity.dsig.XPath2Expression_2 = [XPath expression#1]
com.ibm.wsspi.wssecurity.dsig.XPath2Filter_1 = [intersect]
com.ibm.wsspi.wssecurity.dsig.XPath2Order_1 = [1]
com.ibm.wsspi.wssecurity.dsig.XPath2Expression_2 = [XPath expression#2]
com.ibm.wsspi.wssecurity.dsig.XPath2Filter_2 = [subtract]
com.ibm.wsspi.wssecurity.dsig.XPath2Order_2 = [2]
```

http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform

This algorithm specifies the enhancements to SOAP messaging that provide message integrity and confidentiality.

http://www.w3.org/2002/07/decrypt#XML

This algorithm specifies the W3C decryption transform for XML Signature recommendation.

http://www.w3.org/2000/09/xmldsig#enveloped-signature

This algorithm specifies the W3C recommendation for XML digital signatures.

Configuring consumer signing using JAX-RPC to protect message integrity:

You can configure protect message integrity by configuring signing and key information at the server and application level.

Before you begin

About this task

To protect message integrity, you can:

- Configure the signing information for the consumer binding on the application level or at the server level
- Configure the key information for the consumer binding on the application level or at the server level

Procedure

- To configure the signing information for the consumer binding on the application level, see the steps outlined in “Configuring the signing information using JAX-RPC for the consumer binding on the application level”
- To configure the signing information for the consumer binding on the server level, see the steps outlined in “Configuring the signing information using JAX-RPC for the consumer binding on the server level” on page 3372
- To configure the key information for the consumer binding on the application level, see the steps outlined in “Configuring the key information for the consumer binding on the application level” on page 3324
- To configure the key information for the consumer binding on the server level, see the steps outlined in “Configuring the key information for the consumer binding using JAX-RPC on the server level” on page 3377

Results

By completing the steps in these tasks, you have configured the consumer signing to protect the integrity of messages.

Configuring the signing information using JAX-RPC for the consumer binding on the application level:

You can configure the signing information for the server-side request consumer and the client-side response consumer bindings at the application level.

Before you begin

Note: For WebSphere Application Server version 6.x or earlier only, in the server-side extensions file and the client-side deployment descriptor extensions file, you must specify which parts of the message are signed.

About this task

Configure the key information that is referenced by the key information references on the signing information panel within the administrative console. WebSphere Application Server uses the signing information on the consumer side to verify the integrity of the received SOAP message by validating that the message parts are signed. Complete the following steps to configure the signing information for the server-side request consumer and client-side response consumer sections of the bindings files on the application level.

Procedure

1. Access the administrative console.

   To access the administrative console, enter `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.
2. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
3. Under Manage modules, click ***URI_name***.
4. Under Web Services Security Properties you can access the signing information for the request generator and response generator bindings.
   - To configure the request consumer signing information, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
   - To configure the response consumer signing information, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
5. Under Required properties, click **Signing information**.
6. Click **New** to create a signing information configuration, click **Delete** to delete an existing configuration, or click the name of an existing signing information configuration to edit its settings. If you are creating a new configuration, enter a name in the Signing information name field.
7. Select a signature method algorithm from the Signature method field. The signature method is the algorithm that is used to convert the canonicalized <SignedInfo> element in the binding file into the <SignatureValue> element. The algorithm that is specified for the consumer, which is either the request consumer or the response consumer configuration, must match the algorithm specified for the generator, which is either the request generator or response generator configuration. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2000/09/xmldsig#rsa-sha1
   - http://www.w3.org/2000/09/xmldsig#hmac-sha1
   - http://www.w3.org/2000/09/xmldsig#dsa-sha1

     Do not use this algorithm if you want the configured application to be compliant with the Basic Security Profile (BSP). Any ds:SignatureMethod/@Algorithm element in a signature based on a symmetric key must have a value of http://www.w3.org/2000/09/xmldsig#rsa-sha1 or http://www.w3.org/2000/09/xmldsig#hmac-sha1.
8. Select a canonicalization method from the Canonicalization method field. The canonicalization method algorithm is used to canonicalize the <SignedInfo> element before it is incorporated as part of the digital signature operation. The canonicalization algorithm that you specify for the generator must match the algorithm for the consumer. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/10/xml-exc-c14n#
   - http://www.w3.org/2001/10/xml-exc-c14n#WithComments

- http://www.w3.org/TR/2001/REC-xml-c14n-20010315
- http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments

9. Select a key information signature type from the Key information signature type field. The key information signature type specifies how the <KeyInfo> element in the SOAP message is digitally signed. WebSphere Application Server supports the following signature types:

   None Specifies that the key is not signed.

   Keyinfo
   Specifies that the entire KeyInfo element is signed.

   Keyinfochildelements
   Specifies that the child elements of the KeyInfo element are signed.

   If you do not specify one of the previous signature types, WebSphere Application Server uses keyinfo, by default. The key information signature type for the consumer must match the signature type for the generator.

10. Under Additional properties, click **Key information references**.
    a. Click **New** to create a key information reference or click the name of an existing entry to edit its configuration. The Key information references panel is displayed.
    b. Enter a name in the Name field.
    c. Select a key information reference in the Key information reference field. This reference is the key information configuration name that specifies the key information that is used by this signing information configuration.
11. Return to the Signing information panel. Under Additional properties, click **Part references**. On the Part references panel, you can specify references to the message parts that are defined in the deployment descriptor extensions file.
    a. Click **New** to create a new Part reference or click the name of an existing part reference to edit its configuration. The Part reference panel is displayed.
    b. Enter a name in the Part name field. This name is the name of the required integrity configuration in the deployment descriptor extensions file and specifies the message parts that must be digitally signed.
    c. Select a digest method algorithm from the Digest method algorithm field.

       WebSphere Application Server supports the following pre-configured algorithms:
       - http://www.w3.org/2000/09/xmldsig#sha1
       - http://www.w3.org/2001/04/xmlenc#sha256
       - http://www.w3.org/2001/04/xmlenc#sha512

       If you want to specify a custom algorithm, you must configure the custom algorithm in the Algorithm URI panel before setting the digest method algorithm.
12. Under Additional properties, click **Transforms**.
    a. Click **New** to create a new transform or click the name of an existing transform to edit its configuration.
    b. Enter a name in the Transform name field.
    c. Select a transform algorithm from the Transform algorithm field. WebSphere Application Server supports the following pre-configured algorithms:
       - http://www.w3.org/2001/10/xml-exc-c14n#
       - http://www.w3.org/TR/1999/REC-xpath-19991116

         Do not use this transform algorithm if you want your configured application to be compliant with the Basic Security Profile (BSP). Instead use http://www.w3.org/2002/06/xmldsig-filter2 to ensure compliance.
       - http://www.w3.org/2002/06/xmldsig-filter2

- http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform
- http://www.w3.org/2002/07/decrypt#XML
- http://www.w3.org/2000/09/xmldsig#enveloped-signature

The transform algorithm that you select for the consumer must match the transform algorithm that you select for the generator. For each part reference in the signing information, specify both a digest method algorithm and a transform algorithm.

13. Click **OK**.
14. Click **Save** at the top of the panel to save your configuration.

Results

After completing these steps, you have configured the signing information for the consumer.

What to do next

You must specify a similar signing information configuration for the generator.

Key information references collection:

Use this page to view the key information references that are needed for encryption or signing.

To view this administrative console page on the server level, complete the following steps. On the server level, you can configure the key information references for the default consumer bindings only.

1. Click **Servers > Server Types > WebSphere application servers >** *server_name*.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Consumer Bindings, click either of the following links:
   - Click **Encryption information > *encryption_information_name***.
   - Click **Signing information > *signing_information_name***.
4. Under Additional properties, click **Key information reference**.

To view this administrative console page on the application level, complete the following steps. On the application level, you can configure the key information reference for the consumer bindings only.

1. Click **Applications > Application Types > WebSphere enterprise applications***application_name*.
2. Under Modules, click **Manage modules > *URI_name***.
3. Under Web Services Security properties, you can access the signing information for the following bindings:
   - For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Encryption information**. Click **New** to create a new encryption configuration or click the name of a configuration to modify its settings.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Encryption information**. Click **New** to create a new encryption configuration or click the name of a configuration to modify its settings.

Name:

Specifies the name of the Key information reference.

Key information reference:

Specifies a reference to the message parts that are signed or encrypted.

The value of this field is the name of the <requiredIntegrity> or the <requiredConfidentiality> element in the deployment descriptor.

Key information reference configuration settings:

Use this page to specify a reference to the message parts for signature and encryption that is defined in the deployment descriptors.

To view this administrative console page on the server level for the key information references, complete the following steps. On the server level, you can configure the key information references for the default consumer bindings only.

1. Click **Servers > Server Types > WebSphere application servers >** *server_name*.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Consumer Bindings, click either of the following links:
   - Click **Encryption information >** ***encryption_information_name***.
   - Click **Signing information >** ***signing_information_name***.
4. Under Additional properties, click **Key information references**.

To view this administrative console page on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules >** ***URI_name***.
3. Under Web Services Security Properties, you can access the key information references for the following bindings:
   - For the Response consumer (sender) binding, click **Web services: Client security bindings**. Under Response consumer (sender) binding, click **Edit custom**. Under Required properties, click **Encryption information >** ***encryption_information_name***. Under Additional properties, click **Key information references**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Encryption information >** ***encryption_information_name***. Under Additional properties, click **Key information references**.

Name:

Specifies the name of the key information reference.

Key information reference:

Specifies a reference to the message parts that are signed or encrypted.

The value of this field is the name of the <requiredIntegrity> or the <requiredConfidentiality> element in the deployment descriptor. You can specify a signing key configuration for the following bindings:

Table 293. Key information reference binding configurations. The key is used for signing or encrypting message parts.

Binding name	Server level or application level	Path
Default consumer binding	Server level	1. Click **Servers > Server Types > WebSphere application servers >** *server_name*. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under JAX-RPC Default Consumer Bindings, click **Key information**.
Response consumer (receiver) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** . 2. Under Modules, click **Manage modules >** ***URI_name***. 3. Under Web Services Security Properties, click **Web services: Client security bindings**. 4. Under Response consumer (receiver) binding, click **Edit custom**. 5. Under Required properties, click **Key information**.
Request consumer (receiver) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules >** ***URI_name***. 3. Under Web Services Security Properties, click **Web services: Server security bindings**. 4. Under Request consumer (receiver) binding, click **Edit custom**. 5. Under Required properties, click **Key information**.

Configuring the key information using JAX-RPC for the generator binding on the application level:

The key information is used to specify the configuration needed to generate the key for digital signature and encryption. The signing information and the encryption information configurations can share the key information, so they are both defined at the same level.

Before you begin

Before you begin this task, configure the key locators and the token consumers that are referenced by the Key locator reference and Token reference fields within the key information panel.

About this task

This task provides the steps needed for configuring the key information for the request generator (client side) and the response generator (server side) bindings at the application level.

Complete the following information to configure the key information for the generator binding on the application level:

Procedure

1. Locate the key information configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties you can access the key information for the request generator and response generator bindings.

- For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
- For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.

d. Under Required properties, click **Key information**.

e. Click **New** to create a key information configuration, select the box next to an existing configuration and click **Delete** to delete the configuration, or click the name of an existing signing information configuration to edit its settings. If you are creating a new configuration, enter a name in the Key information name field. For example, you might specify `gen_signkeyinfo`.

2. Select a key information type from the Key information type field. The key information type specifies how to reference the security tokens. WebSphere Application Server supports the following key information types:

Key identifier

The security token is referenced using an opaque value that uniquely identifies the token. The algorithm that is used for generating the <KeyIdentifier> element value depends upon the token type. For example, a hash of the important elements of the security token is used for generating the <KeyIdentifier> element value. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
   <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier ValueType="wsse:X509v3">/62wXO...
      </wsse:KeyIdentifier>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

Key name

The security token is referenced using a name that matches an identity assertion within the token. It is recommended that you do not use this key type as it might result in multiple security tokens that match the specified name. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <ds:KeyName>CN=Group1</ds:KeyName>
</ds:KeyInfo>
```

Security token reference

The security token is directly referenced using Universal Resource Identifiers (URIs). The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference>
      <wsse:Reference URI="#mytoken" />
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

Embedded token

The security token is directly embedded within the <SecurityTokenReference> element. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference>
      <wsse:Embedded wsu:Id="tok1" />
         ...
      </wsse:Embedded>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

X509 issuer name and issuer serial

The security token is referenced by an issuer name and an issuer serial number of an X.509 certificate. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference>
     <ds:X509Data>
        <ds:X509IssuerSerial>
```

```
          <ds:X509IssuerName>CN=Jones, O=IBM, C=US
          </ds:X509IssuerName>
          <ds:X509SerialNumber>1040152879
           </ds:X509SerialNumber>
        </ds:X509IssuerSerial>
     </ds:X509Data>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

Each type of key information is described in the Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS standard, which is located at: http://www.oasis-open.org/home/index.php under Web Services Security.

3. Select a key locator reference from the Key locator reference field. This reference specifies a key locator that WebSphere Application Server uses to locate the keys that are used for digital signature and encryption. Before you can select a key locator, you must have configured a key locator. For more information on configuring a key locator, see the following articles:
   - "Configuring the key locator using JAX-RPC for the generator binding on the application level" on page 3405
   - "Configuring the key locator using JAX-RPC for the consumer binding on the application level" on page 3412
4. Click **Get keys** to view a list of key name references. After you click **Get keys**, the key names that are defined in the <sig_klocator> element are shown in the key name reference menu. If you change the key locator reference, you must click **Get keys** again to display the list of key names associated with the new key locator.
5. Select a key name reference from the Key name reference field. This reference specifies the name of a key that is used for generating a digital signature and for encryption. The list of key names provided comes from the key locator specified with the key locator reference.
6. Select a token reference from the Token reference field. This token reference specifies the name of token generator that is used for processing the security token. However, WebSphere Application Server requires this field only when you select Security token reference or Embedded token in the Key information type field. Before specifying a token reference, you must configure a token generator. For more information on configuring a token generator, see "Configuring token generators using JAX-RPC to protect message authenticity at the application level" on page 3327.
7. Optional: If you select Key identifier as the key information type on this panel, you must specify an encoding method, calculation method, value type namespace URI, and a value type local name.
   a. Select an encoding method from the Encoding method field. The encoding method specifies the encoding format for the key identifier. WebSphere Application Server supports the following encoding methods:
      - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary
      - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#HexBinary
   b. Select a calculation method from the Calculation method field. WebSphere Application Server supports the following calculation methods:
      - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#ITSHA1
      - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#IT60SHA1
   c. Specify a value type namespace Uniform Resource Identifier (URI) in the Namespace URI field. In this field, specify the namespace URI of the value type for a security token that is referenced by the key identifier. When you specify the X.509 certificate token, you do not need to specify this option. If you want to specify another token, you must specify the URI of the qualified name (QName) for value type.
   d. Specify a value type local name. This name is the local name of the value type for a security token that is referenced by the key identifier. When this local name is used in conjunction with the corresponding namespace URI, the information is called the value type qualified name or QName.

When you specify the X.509 certificate token, it is recommended that you use the predefined local names. When you specify the predefined local names, you do not need to specify the namespace URI of the value type. However, if you do not use one of the predefined local names, you must specify both the uniform resource identifier (URI) and the local name. WebSphere Application Server provides the following predefined local names:

X.509 certificate token
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

X.509 certificates in a PKIPath
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1

A list of X509 certificates and CRLs in a PKCS#7
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7

LTPA Lightweight Third-Party Authentication token. When you specify a value type local name of LTPA, you must also specify a namespace URI of http://www.ibm.com/websphere/appserver/tokentype/5.0.2.

LTPA_PROPAGATION
Lightweight Third-Party Authentication propagation token. When you specify a value type local name of LTPA_PROPAGATION, you must also specify a namespace URI of http://www.ibm.com/websphere/appserver/tokentype.

8. Click **OK** and then click **Save** to save the configuration.

Results

You have configured the key information for the generator binding at the application level

What to do next

You must specify a similar key information configuration for the consumer.

Key information collection:

Use this page to view the configurations that are currently available for generating or consuming the key for XML digital signatures and XML encryption.

To view this administrative console page on the server level for the key information references, complete the following steps:

1. Click **Servers > Server Types > WebSphere application servers >** *server_name*.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or the JAX-RPC Default Consumer Bindings, click **Key information**.

To view this administrative console page on the application level for the key information references, complete the following steps.

Note: This option is available on the application level for Version 6 and later applications.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules >** ***URI_name***.

3. Under Web Services Security Properties, you can access the signing information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
   - For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
4. Under Required properties, click **Key information**.

Key information name:

Specifies the name that is given for the key configuration.

Key information class name:

Specifies the class name that is used for the key information type.

Key information type:

Specifies the type of mechanism used to reference the security token. The type corresponds to the class name that is specified in the Key information class name field.

Key information configuration settings:

Use this page to specify the related configuration need to specify the key for XML digital signature or XML encryption.

To view this administrative console page on the server level for the key information references, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings or the JAX-RPC Default Consumer Bindings, click **Key information**.
4. Click **New** to create a new configuration or click the configuration name to modify its contents.

To view this administrative console page on the application level for the key information references, complete the following steps.

Note: This option is available on the application level for Version 6.*x* applications.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Additional properties, you can access the signing information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.

- For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
- For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.

4. Under Required properties, click **Key information**.
5. Click **New** to create a new configuration or click the configuration name to modify its contents.

Before clicking **Properties** under Additional properties, you must enter a value in the **Key information name** field and select an option for the Key information type and Key locator reference options.

Key information name:

Specifies a name for the key information configuration.

Key information type:

Specifies the type of key information. The key information type specifies how to reference security tokens.

This product supports the following types of key information. Each type of key information is described in Web Services Security: SOAP Message Security 1.0 (WS-Security 2004)

Table 294. Key information types. These types of key information are supported by the product.

Type	Description
Key identifier	The security token is referenced using an opaque value that uniquely identifies the token.
Key name	The security token is referenced using a name that matches an identity assertion within the token.
Security token reference	With this type, the security token is directly referenced.
Embedded token	With this type, the security token reference is embedded.
X509 issuer name and issuer serial	With this type, the security token is referenced by an issuer and serial number of an X.509 certificate

The X.509 issuer name and issuer serial is described in Web Services Security: X.509 Certificate Token Profile Version 1.0. The other types are described in Web Services Security: SOAP Message Security 1.0 (WS-Security 2004).

If you select **Key identifier** for the key information type, you can specify values in the following fields on this panel:

- Encoding method
- Calculation method
- Value type namespace URI
- Value type local name

Key locator reference:

Specifies the reference that is used to retrieve the key for digital signature and encryption.

Before specifying a key locator reference, you must configure a key locator. You can specify a signing key configuration for the following bindings:

Table 295. Signing key binding configurations. The key is used during digital signature and encryption.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Key locators**. 4. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Request sender binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Request sender binding, click **Edit**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Response receiver binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Request receiver binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Response sender binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Response sender binding, click **Edit**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Request generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.

Table 295. Signing key binding configurations (continued). The key is used during digital signature and encryption.

Binding name	Server level or application level	Path
Response consumer (receiver) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Request consumer (receiver) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Response generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.

Key name reference:

Specifies the name of the key that is used for generating digital signature and encryption.

This field is displayed for the default generator and is also displayed for the request generator and response generator.

Table 296. Key name reference binding configurations. The key is used during digital signature and encryption.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Key locators**. 4. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.
Request generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.

Table 296. Key name reference binding configurations (continued). The key is used during digital signature and encryption.

Binding name	Server level or application level	Path
Response generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. 4. Under Additional properties, click **Key locators**. 5. Click **New** to create a new key locator or click the name of a configured key locator to modify its configuration.

Token reference:

Specifies the name of a token generator or token consumer that is used for processing a security token.

The application server requires this field only when you specify Security token reference or Embedded token in the **Key information type** field. The **Token reference** field is also required when you specify a key identifier type for the consumer. Before specifying a token reference, you must configure a token generator or token consumer. You can specify a token configuration for the following bindings on the following levels:

Table 297. Token reference binding configurations. The reference information is used for a security token reference or an embedded token.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under JAX-RPC Default Generator Bindings, click **Token generator**. 4. Click **New** to create a new token generator or click the name of a configured token generator to modify its configuration.
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under JAX-RPC Default Consumer Bindings, click **Token consumer**. 4. Click **New** to create a new token consumer or click the name of a configured token consumer to modify its configuration.
Request generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**. 4. Under Additional properties, click **Token generators**. 5. Click **New** to create a new token generator or click the name of a configured token generator to modify its configuration.

Table 297. Token reference binding configurations (continued). The reference information is used for a security token reference or an embedded token.

Binding name	Server level or application level	Path
Response consumer (receiver) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. 4. Under Required properties, click **Token consumers**. 5. Click **New** to create a new token consumer or click the name of a configured token consumer to modify its configuration.
Request consumer (receiver) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**. 4. Under Required properties, click **Token consumers**. 5. Click **New** to create a new token consumer or click the name of a configured token consumer to modify its configuration.
Response generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. 4. Under Additional properties, click **Token generators**. 5. Click **New** to create a new token generator or click the name of a configured token generator to modify its configuration.

Encoding method:

Specifies the encoding method that indicates the encoding format for the key identifier.

This field is valid when you specify Key identifier in the Key information type field. This product supports the following encoding methods:

- http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary
- http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#HexBinary

This field is available for the default generator binding only.

Calculation method:

This field is valid when you specify Key identifier in the **Key information type** field. This product supports the following calculation methods:

- http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#ITSHA1
- http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#IT60SHA1

This field is available for the generator binding only.

Value type namespace URI:

Specifies the namespace Uniform Resource Identifier (URI) of the value type for a security token that is referenced by the key identifier.

This field is valid when you specify Key identifier in the **Key information type** field. When you specify the X.509 certificate token, you do not need to specify this option. If you want to specify another token, specify the URI of `QName` for value type.

This product provides the following predefined value type URIs for the Lightweight Third Party Authentication (LTPA) token:

- http://www.ibm.com/websphere/appserver/tokentype
- http://www.ibm.com/websphere/appserver/tokentype/5.0.2

This field is available for the generator binding only.

Value type local name:

Specifies the local name of the value type for a security token that is referenced by the key identifier.

When this local name is used with the corresponding namespace URI, the information is called the *value type qualified name* or *QName*.

This field is valid when you specify Key identifier in the **Key information type** field. When you specify the X.509 certificate token, it is recommended that you use the predefined local names. When you specify the predefined local names, you do not need to specify the URI of the value type. This product provides the following predefined local names:

X.509 certificate token
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

X.509 certificates in a PKIPath
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1

A list of X509 certificates and CRLs in a PKCS#7
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7

Lightweight Third Party Authentication (LTPA)

LTPA_PROPAGATION

Attention: For LTPA, the value type local name is `LTPA`. If you enter `LTPA` for the local name, you must specify the `http://www.ibm.com/websphere/appserver/tokentype/5.0.2` URI value in the **Value type URI** field as well. For LTPA token propagation, the value type local name is `LTPA_PROPAGATION`. If you enter `LTPA_PROPAGATION` for the local name, you must specify the `http://www.ibm.com/websphere/appserver/tokentype` URI value in the **Value type URI** field as well. For the other predefined value types (User name token, X509 certificate token, X509 certificates in a PKIPath, and a list of X509 certificates and CRLs in a PKCS#7), the value for the **Value type local name** field begins with `http://`. For example, if you are specifying the user name token for the value type, enter `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken` in the **Value type local name** field and then you do not need to enter a value in the value type URI field.

When you specify a custom value type for custom tokens, you can specify the local name and the URI of the quality name (QName) of the value type. For example, you might specify `Custom` for the local name and `http://www.ibm.com/custom` for the URI.

This field is also available for the generator binding only.

Configuring the key information for the consumer binding on the application level:

You can configure the key information for the request consumer (server side) and the response consumer (client side) bindings at the application level.

Before you begin

Configure the key locators and the token consumers that are referenced by the Key locator reference and the Token reference fields within the key information panel.

About this task

This task provides the steps that are needed for configuring the key information for the request consumer (server side) and the response consumer (client side) bindings at the application level. The key information on the consumer side is used for specifying the information about the key, which is used for validating the digital signature in the received message or for decrypting the encrypted parts of the message. Complete the following steps to configure the key information for consumer binding on the application level.

Procedure

1. Locate the key information configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties, you can access the key information for the request consumer and response consumer bindings.
      - For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under request consumer (receiver) binding, click **Edit custom**.
      - For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under response consumer (receiver) binding, click **Edit custom**.
   d. Under Required properties, click **Key information**.
   e. Click one of the following to work with key information configuration:

      New To create a key information configuration. Enter a name in the Key information name field. For example, you might specify `con_signkeyinfo`.

      Delete To delete a configuration (selected in the box next to that configuration).

      an existing key information configuration
      To edit the settings of a key information configuration.
2. Select a key information type from the Key information type field. The key information types specify different mechanisms for referencing security tokens using the <wsse:SecurityTokenReference> element within the <ds:KeyInfo> element. WebSphere Application Server supports the following key information types:

   Key identifier
   The security token is referenced using an opaque value that uniquely identifies the token. The algorithm that is used for generating the <KeyIdentifier> element value depends upon the token type. For example, you can use the identifier for the public keys that are defined in the Internet Engineering Task Force (IETF) Request for Comment (RFC) 3280. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
   <wsse:SecurityTokenReference>
      <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/2004/01
      /oasis-200401-wss-x509-token-profile-1.0#X509v3SubjectKeyIdentifier">
      /62wXO...
      </wsse:KeyIdentifier>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

   Key name
   The security token is referenced using a name that matches an identity assertion within the token. It is recommended that you do not use this key type as it might result in multiple

security tokens that match the specified name. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <ds:KeyName>CN=Group1</ds:KeyName>
</ds:KeyInfo>
```

In general, use a key name when you use a Key-Hashing Message Authentication Code (HMAC) digital signature algorithm, such as http://www.w3.org/2000/09/xmldsig#hmac-sha1.

Security token reference

The security token is directly referenced using Universal Resource Identifiers (URIs). The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference>
      <wsse:Reference URI='#SomeCert'
       ValueType="http://docs.oasis-open.org/wss/2004/01/
          oasis-200401-wss-x509-token-profile-1.0#X509v3" />
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

Attention: As stated in the Web Services Interoperability Organization (WS-I) Basic Security Profile Version 1 draft and shown in the previous example, the wsse:Reference element in a SECURE_ENVELOPE must have a ValueType attribute.

Embedded token

The security token is directly embedded within the <SecurityTokenReference> element. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference>
      <wsse:Embedded wsu:Id="tok1" />
         ...
      </wsse:Embedded>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

X509 issuer name and issuer serial

The security token is referenced by an issuer name and an issuer serial number of an X.509 certificate. The following <KeyInfo> element is generated in the SOAP message for this key information type:

```
<ds:KeyInfo>
   <wsse:SecurityTokenReference>
     <ds:X509Data>
        <ds:X509IssuerSerial>
           <ds:X509IssuerName>CN=Jones, O=IBM, C=US</ds:X509IssuerName>
           <ds:X509SerialNumber>1040152879</ds:X509SerialNumber>
        </ds:X509IssuerSerial>
     </ds:X509Data>
   </wsse:SecurityTokenReference>
</ds:KeyInfo>
```

Each type of key information is described in the Web Services Security: SOAP Message Security 1.0 (WS-Security 2004) OASIS standard, which is located at: http://www.oasis-open.org/home/index.php under Web Services Security.

3. Select a key locator reference from the Key locator reference field. The value of this field is a reference to a key locator that WebSphere Application Server uses to locate the keys that are used for digital signature and encryption. Before you can select a key locator, you must configure a key locator. For more information on configuring a key locator, see “Configuring the key locator using JAX-RPC for the consumer binding on the application level” on page 3412.
4. Select a token reference from the Token reference field. The token reference specifies a reference to a token consumer that is used for processing the security token in the message. However, WebSphere Application Server requires this field only when you select Security token reference or Embedded token in the Key information type field. Before specifying a token reference, you must configure a

token consumer. For more information on configuring a token consumer, see “Configuring token consumers using JAX-RPC to protect message authenticity at the application level” on page 3344.

Select **(none)** if a token consumer is not required for this key information configuration.

5. Click **OK** and **Save** to save this configuration.

Results

You have configured the key information for the request or response (or both) consumer binding at the application level.

What to do next

If you have not configured the key information for the generator binding, you must specify a similar key information configuration for the generator. After you configure the key information for both the consumer and the generator, configure the signing information or encryption information, which references the key information that is specified in this key information task.

Configuring token generators using JAX-RPC to protect message authenticity at the application level:

When you specify the token generators at the application level, the information is used on the generator side to generate the security token.

Before you begin

You need to understand that the keystore/alias information that you provide for the generator, and the keystore/alias information that you provide for the consumer are used for different purposes. The main difference applies to the Alias for an X.509 callback handler:

Generator

When used in association with an ecryption generator, the alias supplied for the generator is used to retrieve the public key to encrypt the message. A password is not required. The alias that is entered on a callback handler associated with an encryption generator must be accessible without a password. This means that the alias must not have private key information associated with it in the keystore. When used in association with a signature generator, the alias supplied for the generator is used retrieve the private key to sign the message. A password is required.

Consumer

When used in association with an ecryption consumer, the alias supplied for the consumer is used retrieve the private key to decrypt the message. A password is required.

When associated with a signature consumer, the alias supplied for the consumer is used strictly to retrieve the public key that is used to resolve an X.509 certificate that is not passed in the SOAP security header as a BinarySecurityToken. A password is not required.

About this task

Complete the following steps to configure the token generator on the application level:

Procedure

1. Locate the token generator panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties you can access the token generators for the following bindings:

- For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
- For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.

d. Under Additional properties, click **Token generators**.

e. Click **New** to create a token generator configuration, select an existing configuration. Click **Delete** to delete an existing configuration, or click the name of an existing token generator configuration to edit its settings. If you are creating a new configuration, enter a unique name in the **Token generator name** field. For example, you might specify `gen_signtgen`.

2. Specify a class name in the **Token generator class name** field. The token generator class must implement the com.ibm.wsspi.wssecurity.token.TokenGeneratorComponent interface. The token generator class name for the request generator and the response generator must be similar to the token consumer class name for the request consumer and the response consumer. For example, if your application requires a username token consumer, you can specify the com.ibm.wsspi.wssecurity.token.UsernameTokenConsumer class name on the token consumer panel for the application level and the com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator class name in this field.
3. Optional: Select a part reference in the **Part reference** field. The part reference indicates the name of the security token that is defined in the deployment descriptor.

   Important: On the application level, if you do not specify a security token in your deployment descriptor, the **Part reference** field is not displayed. If you define a security token called `user_tgen` in your deployment descriptor, `user_tgen` is displayed as an option in the **Part reference** field. You can specify a security token in the deployment descriptor when you assemble your application using an assembly tool.
4. Select either **None** or **Dedicated signing information** for the certificate path. Select **None** when the token generator does not use the PKCS#7 token type. When the token generator uses the PKCS#7 token type and you want to package certificate revocation lists (CRLs) in the security token, select **Dedicated signing information** and select a certificate store. To configure a collection certificate store and certificate revocation lists for the generator bindings on the application level, complete the following steps:

   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.

   b. Under Related Items, click **EJB Modules** or **Web Modules** > ***URI_name***.

   c. Under Additional Properties you can access the collection certificate store configuration for the following bindings:
   - For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.

   d. Under Additional properties, click **Collection certificate store**.

   also see the information about configuring a collection certificate store.
5. Optional: Select the **Add nonce** option. This option indicates whether a nonce is included in the user name token for the token generator. Nonce is a unique, cryptographic number that is embedded in a message to help stop repeat, unauthorized attacks of user name tokens. The **Add nonce** option is valid only when the generated token type is a user name token and is available only for the request generator binding.

   If you select the **Add nonce** option, you can specify the following properties under Additional properties. These properties are used by the request consumer.

Table 298. Additional nonce properties. Use the nonce properties to include nonce in the user name token.

Property name	Default value	Explanation
com.ibm.ws.wssecurity.config.token. BasicAuth.Nonce.cacheTimeout	600 seconds	Specifies the timeout value, in seconds, for the nonce value that is cached on the server.
com.ibm.ws.wssecurity.config.token. BasicAuth.Nonce.clockSkew	0 seconds	Specifies the time, in seconds, before the nonce time stamp expires.
com.ibm.ws.wssecurity.config.token. BasicAuth.Nonce.maxAge	300 seconds	Specifies the clock skew value, in seconds, to consider when WebSphere Application Server checks the timeliness of the message.

On the server level, you can specify these additional properties for a nonce on the Default bindings for Web Services Security panel within the administrative console. To access the panel, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

6. Optional: Select the **Add timestamp** option. This option indicates whether to insert a time stamp into the user name token. The **Add timestamp** option is valid only when the generated token type is a user name token and is available only for the request generator binding.
7. Specify the value type local name in the **Local name** field. For a user name token and an X.509 certificate security token, WebSphere Application Server provides predefined local names for the value type. When you specify any of the following local names, you do not need to specify a value type URI:

   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken
   This local name specifies a user name token.

   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3
   This local name specifies an X.509 certificate token.

   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1
   This local name specifies X.509 certificates in a public key infrastructure (PKI) path.

   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7
   This local name specifies a list of X.509 certificates and certificate revocation lists in a PKCS#7 format.

   For an LTPA token, you can use `LTPA` for the value type local name and `http://www.ibm.com/websphere/appserver/tokentype/5.0.2` for the value type Uniform Resource Identifier (URI). For LTPA token propagation, you can use `LTPA_PROPAGATION` for the value type local name and `http://www.ibm.com/websphere/appserver/tokentype` for the value type URI.
8. Optional: Specify the value type URI in the **URI** field. This entry specifies the namespace URI of the value type for the generated token.
9. Click **OK** and **Save** to save the configuration.
10. Click the name of your token generator configuration.
11. Under Additional properties, click **Callback handler**.
12. Specify the settings for the callback handler.
    a. Specify a class name in the **Callback handler class name** field. This class name is the name of the callback handler implementation class that is used to plug-in a security token framework. The specified callback handler class must implement the javax.security.auth.callback.CallbackHandler interface and must provide a constructor using the following syntax:

```
MyCallbackHandler(String username, char[] password, java.util.Map properties)
```

Where:

username
Specifies the user name that is passed into the configuration.

password
Specifies the password that is passed into the configuration.

properties
Specifies the other configuration properties that are passed into the configuration.

This constructor is required if the callback handler needs a user name and a password. However, if the callback handler does not need a user name and a password, such as X509CallbackHandler, use a constructor with the following syntax:

```
MyCallbackHandler(java.util.Map properties)
```

WebSphere Application Server provides the following default callback handler implementations:

com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler
This callback handler uses a login prompt to gather the user name and password information. However, if you specify the user name and password on this panel, a prompt is not displayed and WebSphere Application Server returns the user name and password to the token generator. Use this implementation for a Java Platform, Enterprise Edition (Java EE) application client only. If you use this implementation, you must provide a basic authentication user ID and password on this panel.

com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler
This callback handler does not issue a prompt and returns the user name and password if it is specified on this panel. You can use this callback handler when the web service is acting as a client. If you use this implementation, you must provide a basic authentication user ID and password on this panel.

com.ibm.wsspi.wssecurity.auth.callback.StdinPromptCallbackHandler
This callback handler uses a standard-in prompt to gather the user name and password. However, if the user name and password is specified on this panel, WebSphere Application Server does not issue a prompt, but returns the user name and password to the token generator. Use this implementation for a Java Platform, Enterprise Edition (Java EE) application client only. If you use this implementation, you must provide a basic authentication user ID and password on this panel.

com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler
This callback handler is used to obtain the Lightweight Third Party Authentication (LTPA) security token from the Run As invocation Subject. This token is inserted in the Web Services Security header within the SOAP message as a binary security token. However, if the user name and password are specified on this panel, WebSphere Application Server authenticates the user name and password to obtain the LTPA security token rather than obtaining it from the Run As Subject. Use this callback handler only when the web service is acting as a client on the application server. It is recommended that you do not use this callback handler on a Java EE application client. If you use this implementation, you must provide a basic authentication user ID and password on this panel.

com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler
This callback handler is used to create the X.509 certificate that is inserted in the Web Services Security header within the SOAP message as a binary security token. A keystore and a key definition is required for this callback handler. If you use this implementation, you must provide a key store password, path, and type on this panel.

com.ibm.wsspi.wssecurity.auth.callback.PKCS7CallbackHandler
This callback handler is used to create X.509 certificates encoded with the PKCS#7 format. The certificate is inserted in the Web Services Security header in the SOAP message as a binary security token. A keystore is required for this callback handler. You can specify a certificate revocation list (CRL) in the collection certificate store. The CRL is

encoded with the X.509 certificate in the PKCS#7 format. If you use this implementation, you must provide a key store password, path, and type on this panel.

com.ibm.wsspi.wssecurity.auth.callback.PkiPathCallbackHandler
This callback handler is used to create X.509 certificates encoded with the PkiPath format. The certificate is inserted in the Web Services Security header within the SOAP message as a binary security token. A keystore is required for this callback handler. A CRL is not supported by the callback handler; therefore, the collection certificate store is not required or used. If you use this implementation, you must provide a key store password, path, and type on this panel.

The callback handler implementation obtains the required security token and passes it to the token generator. The token generator inserts the security token in the Web Services Security header within the SOAP message. Also, the token generator is a plug-in point for the pluggable security token framework. Service providers can provide their own implementation, but the implementation must use the com.ibm.wsspi.wssecurity.token.TokenGeneratorComponent interface.

b. Optional: Select the **Use identity assertion** option. Select this option if you have identity assertion defined in the IBM extended deployment descriptor. This option indicates that only the identity of the initial sender is required and inserted into the Web Services Security header within the SOAP message. For example, WebSphere Application Server sends only the user name of the original caller for a username token generator. For an X.509 token generator, the application server sends the original signer certification only.

c. Optional: Select the **Use RunAs identity** option. Select this option if you have identity assertion defined in the IBM extended deployment descriptor and you want to use the Run As identity instead of the initial caller identity for identity assertion in a downstream call. This option is valid only if you have configured Username TokenGenerator as a token generator.

d. Optional: Specify the basic authentication user ID in the **Basic authentication user ID** field. This entry specifies the user name that is passed to the constructors of the callback handler implementation. The basic authentication user name and password are used if you specified one of the following default callback handler implementations in the **Callback handler class name** field:
   - com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.StdinPromptCallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler

e. Optional: Specify the basic authentication password in the **Basic authentication password** field. This entry specifies the password that is passed to the constructors of the callback handler implementation.

f. Optional: Specify the key store password in the **Key store password** field. This entry specifies the password used to access the key store file. The key store and its configuration are used if you select on of the following default callback handler implementations that are provided by WebSphere Application Server:

   com.ibm.wsspi.wssecurity.auth.callback.PKCS7CallbackHandler
   The keystore is used to build the X.509 certificate with the certificate path.

   com.ibm.wsspi.wssecurity.auth.callback.PkiPathCallbackHandler
   The keystore is used to build the X.509 certificate with the certificate path.

   com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler
   The keystore is used to retrieve the X.509 certificate.

g. Optional: Specify the key store path in the **Path** field. It is recommended that you use the `${USER_INSTALL_ROOT}` in the path name as this variable expands to the WebSphere Application Server path on your machine. To change the path used by this variable, click **Environment** > **WebSphere variables**, and click **USER_INSTALL_ROOT**. This field is required when you use the

com.ibm.wsspi.wssecurity.auth.callback.PKCS7CallbackHandler, com.ibm.wsspi.wssecurity.auth.callback.PkiPathCallbackHandler, or com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler callback handler implementations.

h. Optional: Select the key store type in the **Type** field. This selection indicates the format used by the keystore file. You can select one of the following values for this field:

JKS Use this option if the keystore uses the Java Keystore (JKS) format.

JCEKS
Use this option if the Java Cryptography Extension is configured in the software development kit (SDK). The default IBM JCE is configured in WebSphere Application Server. This option provides stronger protection for stored private keys by using Triple DES encryption.

JCERACFKS
Use JCERACFKS if the certificates are stored in a SAF key ring (z/OS only).

PKCS11KS (PKCS11)
Use this format if your keystore uses the PKCS#11 file format. Keystores using this format might contain RSA keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

PKCS12KS (PKCS12)
Use this option if your keystore uses the PKCS#12 file format.

13. Click **OK** and then click **Save** to save the configuration.
14. Click the name of your token generator configuration.
15. Under Additional properties, click **Callback handler** > **Keys**.
16. Specify the key name, key alias, and the key password.
    a. Click **New** to create a key configuration, click **Delete** to delete an existing configuration, or click the name of an existing key configuration to edit its settings. If you are creating a new configuration, enter a unique name in the **Key name** field. For digital signatures, the key name is used by the request generator or response generator signing information to determine which key is used to digitally sign the message. For encryption, the key name is used to determine the key used for encryption. The key name must be a fully qualified, distinguished name. For example, CN=Bob,O=IBM,C=US.
    b. Specify the key alias in the **Key alias** field. The key alias is used by the key locator to find the key within the keystore file.
    c. Specify the key password in the **Key password** field. This password is needed to access the key object within the keystore file.
17. Click **OK** and **Save** to save the configuration.

Results

You have configured the token generator for the application level.

What to do next

You must specify a similar token consumer configuration for the application level.

Request generator (sender) binding configuration settings:

Use this page to specify the binding configuration for the request generator.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** .
2. Under Modules, click **Manage modules**.
3. Click the Uniform Resource Identifier (URI).
4. Under Web Services Security Properties, click **Web services: Client security bindings**.
5. Under Request generator (sender) binding, click **Edit custom**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

The security constraints or bindings are defined using the application assembly process before the application is installed.

This product provides assembly tools to assemble your application.

If the security constraints are defined in the application, you must either define the corresponding binding information or select the Use defaults option on this panel and use the default binding information for the server level. The default binding provided by this product is a sample. Do not use this sample in a production environment without modifying the configuration. The security constraints define what is signed or encrypted in the Web Services Security message. The bindings define how to enforce the requirements.

Digital signature security constraint (integrity)

The following table shows the required and optional binding information when the digital signature security constraint (integrity) is defined in the deployment descriptor.

Table 299. Binding information for digital signature security constraints. The binding information is used for digitally signing messages.

Information type	Required or optional
Signing information	Required
Key information	Required
Key locators	Optional
Collection certificate store	Optional
Token generator	Optional
Properties	Optional

You can use the key locators and the collection certificate store that are defined at the server-level.

Encryption constraint (confidentiality)

The following table shows the required and optional binding information when the encryption constraint (confidentiality) is defined in the deployment descriptor.

Table 300. Binding information for encryption constraints. The binding information is used for encrypting messages.

Information type	Required or optional
Encryption information	Required
Key information	Required
Key locators	Optional
Collection certificate store	Optional
Token generator	Optional
Properties	Optional

You can use the key locators and the collection certificate store that are defined at the server-level.

Security token constraint

The following table shows the required and optional binding information when the security token constraint is defined in the deployment descriptor.

Table 301. Binding information for security token constraints. The binding information is used for signing or encrypting messages.

Information type	Required or optional
Token generator	Required
Collection certificate store	Optional
Properties	Optional

You can use the collection certificate store that is defined at the server-level.

Use defaults:

Select this option if you want to use the default binding information from the server-level.

Component:

Specifies the enterprise bean in an assembled EJB module.

Port:

Specifies the port in the web service that is defined during application assembly.

Web service:

Specifies the name of the web service that is defined during application assembly.

Response generator (sender) binding configuration settings:

Use this page to specify the binding configuration for the response generator or response sender.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules**.
3. Click the Uniform Resource Identifier (URI).
4. Under Web Services Security Properties, click **Web services: Server security bindings**.
5. Under Response generator (sender) binding, click **Edit custom**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

The security constraints or bindings are defined using the application assembly process before the application is installed.

This product provides assembly tools to assemble your application.

If the security constraints are defined in the application, you must either define the corresponding binding information or select the **Use defaults** option on this panel and use the default binding information for the server-level. The default binding that is provided by this product is a sample. Do not use this sample in a

production environment without modifying the configuration. The security constraints define what is signed or encrypted in the Web Services Security message. The bindings define how to enforce the requirements.

Digital signature security constraint (integrity)

The following table shows the required and optional binding information when the digital signature security constraint (integrity) is defined in the deployment descriptor.

Table 302. Binding information for digital signature constraints. The binding information is used for digitally signing messages.

Information type	Required or optional
Signing information	Required
Key information	Required
Key locators	Optional
Collection certificate store	Optional
Token generator	Optional
Properties	Optional

You can use the key locators and the collection certificate store that are defined at the server-level.

Encryption constraint (confidentiality)

The following table shows the required and optional binding information when the encryption constraint (confidentiality) is defined in the deployment descriptor.

Table 303. Binding information for encryption constraints. The binding information is used for encrypting messages.

Information type	Required or optional
Encryption information	Required
Key information	Required
Key locators	Optional
Collection certificate store	Optional
Token generator	Optional
Properties	Optional

You can use the key locators and the collection certificate store that are defined at the server-level.

Security token constraint

The following table shows the required and optional binding information when the security token constraint is defined in the deployment descriptor.

Table 304. Binding information for security token constraints. The binding information is used for signing and encrypting messages.

Information type	Required or optional
Token generator	Required
Collection certificate store	Optional
Properties	Optional

You can use the collection certificate store that is defined at the server-level.

Use defaults:

Select this option if you want to use the default binding information from the server level.

Port:

Specifies the port number in the web service that is defined during application assembly.

Web service:

Specifies the name of the web service that is defined during application assembly.

Callback handler configuration settings for JAX-WS:

Use this page to specify how to acquire the security token that is inserted in the Web Services Security header for JAX-WS within the SOAP message. The token acquisition is a pluggable framework that leverages the Java Authentication and Authorization Service (JAAS) `javax.security.auth.callback.CallbackHandler` interface for acquiring the security token.

To view this administrative console page for the callback handler on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** .
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default generator bindings, click **Token generators >** ***token_generator_name***.
4. Under Additional properties, click **Callback handler**.

To view this administrative console page for the callback handler on the application level , complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage Modules*****URI_name***.
3. Under Web Services Security properties, you can access the callback handler information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**. Under Additional properties, click **Token generator**. Click **New** to create a new token generator configuration or click the name of an existing configuration to modify its settings. Under Additional properties, click **Callback handler**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. Under Additional properties, click **Token generator**. Click **New** to create a new token generator configuration or click the name of an existing configuration to modify its settings. Under Additional properties, click **Callback handler**.

Callback handler class name:

Specifies the name of the callback handler implementation class that is used to plug in a security token framework.

The specified callback handler class must implement the `javax.security.auth.callback.CallbackHandler` class. The implementation of the JAAS `javax.security.auth.callback.CallbackHandler` interface must provide a constructor using the following syntax:

```
MyCallbackHandler(String username, char[] password,
    java.util.Map properties)
```

Where:

username
Specifies the user name that is passed into the configuration.

password
Specifies the password that is passed into the configuration.

properties
Specifies the other configuration properties that are passed into the configuration.

The application server provides the following default callback handler implementations:

com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler
This callback handler uses a login prompt to gather user name and password information. However, if you specify the user name and password on this panel, a prompt is not displayed and the application server returns the user name and password to the token generator if it is specified on this panel. Use this implementation for a Java Platform, Enterprise Edition (Java EE) application client only.

com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler
This callback handler does not issue a prompt and returns the user name and password if it is specified on this panel. You can use this callback handler when the web service is acting as a client.

com.ibm.wsspi.wssecurity.auth.callback.StdinPromptCallbackHandler
This callback handler uses a standard-in prompt to gather the user name and password. However, if the user name and password is specified on this panel, the application server does not issue a prompt, but returns the user name and password to the token generator. Use this implementation for a Java Platform, Enterprise Edition (Java EE) application client only.

com.ibm.wsspi.wssecurity.auth.callback.StdinPromptCallbackHandler
This callback handler uses a standard-in prompt to gather the user name and password. However, if the user name and password is specified on this panel, the application server does not issue a prompt, but returns the user name and password to the token generator. Use this implementation for a Java Platform, Enterprise Edition (Java EE) application client only.

com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler
This callback handler is used to obtain the Lightweight Third Party Authentication (LTPA) security token from the RunAs invocation Subject. This token is inserted in the Web Services Security header within the SOAP message as a binary security token. However, if the user name and password are specified on this panel, the application server authenticates the user name and password to obtain the LTPA security token rather than obtaining it from the RunAs Subject. Use this callback handler only when the web service is acting as a client on the application server. It is recommended that you do not use this callback handler on a Java EE application client.

com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler
This callback handler is used to create the X.509 certificate that is inserted in the Web Services Security header within the SOAP message as a binary security token. A keystore and a key definition is required for this callback handler.

com.ibm.wsspi.wssecurity.auth.callback.PKCS7CallbackHandler
This callback handler is used to create X.509 certificates encoded with the PKCS#7 format. The certificate is inserted in the Web Services Security header in the SOAP message as a binary security token. A keystore is required for this callback handler. You must specify a certificate revocation list (CRL) in the collection certificate store. The CRL is encoded with the X.509 certificate in the PKCS#7 format.

com.ibm.wsspi.wssecurity.auth.callback.PkiPathCallbackHandler
This callback handler is used to create X.509 certificates encoded with the PkiPath format. The certificate is inserted in the Web Services Security header within the SOAP message as a binary security token. A keystore is required for this callback handler. A CRL is not supported by the callback handler; therefore, the collection certificate store is not required or used.

The callback handler implementation obtains the required security token and passes it to the token generator. The token generator inserts the security token in the Web Services Security header within the SOAP message. Also, the token generator is the plug-in point for the pluggable security token framework. Service providers can provide their own implementation, but the implementation must use the `com.ibm.websphere.wssecurity.wssapi.token.SecurityToken` interface. The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to create the security token on the generator side and to validate (authenticate) the security token on the consumer side, respectively.

Use identity assertion:

Select this option if you have identity assertion defined in the IBM extended deployment descriptor.

This option indicates that only the identity of the initial sender is required and inserted into the Web Services Security header within the SOAP message. For example, the application server sends only the user name of the original caller for a Username TokenGenerator. For an X.509 token generator, the application server sends the original signer certification only.

Use RunAs identity:

Select this option if you have identity assertion defined in the IBM extended deployment descriptor and you want to use the Run As identity instead of the initial caller identity for identity assertion for a downstream call.

This option is valid only if you have Username TokenGenerator configured as a token generator.

Basic authentication user ID:

Specifies the user name that is passed to the constructors of the callback handler implementation.

The basic authentication user name and password are used if you select one of the following default callback handler implementations provided by this product:

- `com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler`
- `com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler`
- `com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler`
- `com.ibm.wsspi.wssecurity.auth.callback.StdinPromptCallbackHandler`

These implementations are described in detail under the **Callback handler class name** field description in this article.

Basic authentication password:

Specifies the password that is passed to the constructor of the callback handler.

The keystore and its related configuration are used if you select one of the following default callback handler implementations provided by this product:

com.ibm.wsspi.wssecurity.auth.callback.PKCS7CallbackHandler
The keystore is used to build the X.509 certificate with the certificate path.

com.ibm.wsspi.wssecurity.auth.callback.PkiPathCallbackHandler
The keystore is used to build the X.509 certificate with the certificate path.

com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler
The keystore is used to retrieve the X.509 certificate.

Keystore: Select **None** if no keystore is needed for this configuration.

Select **Predefined keystore** to choose predefined keystores with keystore configuration name.

Select **User-defined keystore** to use user-defined keystores.

The following information needs to be specified:

Key store configuration name:

Specifies the name of the key store configuration defined in the keystore settings in secure communications.

Key store password:

Specifies the password that is used to access the keystore file.

Key store path:

Specifies the location of the keystore file.

Use ${*USER_INSTALL_ROOT*} in the path name because this variable expands to the product path on your machine. To change the path used by this variable, click **Environment** > **WebSphere variables** and click **USER_INSTALL_ROOT**.

Key store type:

Specifies the type of keystore file format

Choose one of the following values for this field:

JKS Use this option if the keystore uses the Java Keystore (JKS) format.

JCEKS
Use this option if the Java Cryptography Extension is configured in the software development kit (SDK). The default IBM JCE is configured in the application server. This option provides stronger protection for stored private keys by using Triple DES encryption.

PKCS11KS (PKCS11)
Use this option if your keystore file uses the PKCS#11 file format. Keystore files that use this format might contain Rivest Shamir Adleman (RSA) keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

PKCS12KS (PKCS12)
Use this option if your keystore file uses the PKCS#12 file format.

Key collection:

Use this page to view a list of logical names that is mapped to a key alias in the keystore file.

To view this administrative console page for the key locator collection on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** .
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default generator bindings, click **Token Generators** > ***token_generator_name*** .
4. Under Additional properties, click **Callback handler** > **Keys**.

Keys are also available from the JAX-WS and JAX-RPC security runtime panel by clicking **Key locators** > ***key_locator_name***. Under Additional properties, click **Keys**.

To use this administrative console page for the key locator collection on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access key locators for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
4. Under Additional properties, you can access key locators for the following bindings:
   - For the Request sender, click **Web services: Client security bindings**. Under Request sender binding, click **Edit** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Request receiver, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Response sender, click **Web services: Server security bindings**. Under Response sender binding, click **Edit** > **Key locators**. Under Additional properties, click `Keys`.
   - For the Response receiver, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit** > **Key locators**. Under Additional properties, click **Keys**.

Key name:

Specifies the name of the key object that is found in the keystore file.

Key alias:

Specifies an alias for the key object.

The alias is used when the key locator searches for the key objects in the keystore file.

Key configuration settings:

Use this page to define the mapping of a logical name to a key alias in a keystore file.

To view this administrative console page for the key locator collection on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

3. Under JAX-RPC Default generator bindings, click **Token Generators** > ***token_generator_name***.
4. Under Additional properties, click **Callback handler** > **Keys**.
5. Specify a new key configuration by clicking **New** or by clicking the key configuration name to modify the settings.

Keys are also available from the JAX-WS and JAX-RPC security runtime panel by clicking **Key locators** > ***key_locator_name***. Under Additional properties, click **Keys** > **New**. Specify a new key configuration by clicking New or by clicking the key configuration name to modify the settings.

To use this administrative console page for the key locator collection on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Additional properties, you can access key locators for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Key locators**. Under Additional properties, click **Keys**.
4. Under Web Services Security Properties, you can access key locators for the following bindings:
   - For the Request sender, click **Web services: Client security bindings**. Under Request sender binding, click **Edit** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Request receiver, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Response sender, click **Web services: Server security bindings**. Under Response sender binding, click **Edit** > **Key locators**. Under Additional properties, click **Keys**.
   - For the Response receiver, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit** > **Key locators**. Under Additional properties, click **Keys**.
5. Specify a new key configuration by clicking **New** or by clicking the key configuration name to modify the settings.

Key name:

Specifies the name of the key object. For digital signatures, the key name is used by the request sender or request generator signing information to determine which key is used to digitally sign the message. For encryption, the key name is used to determine the key used for encryption.

The key name must be a fully qualified, distinguished name. For example, `CN:Bob,O=IBM,C=US`.

Note: If you enter the distinguished name with spaces before or after commas and equal symbols, the application server normalizes the distinguished names automatically during run time by removing these extra spaces.

Key alias:

Specifies the alias for the key object, which is used by the key locator to find the key within the keystore file.

Key password:

Specifies the password that is needed to access the key object within the keystore file.

Web services: Client security bindings collection:

Use this page to view a list of application-level, client-side binding configurations for Web Services Security. These bindings are used when a web service is a client to another web service.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, click **Web services: Client security bindings**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Component:

Specifies the enterprise bean in an assembled Enterprise JavaBeans (EJB) module.

Port:

Specifies the port that is used to send messages to a server and receive messages from a server.

Web service:

Specifies the name of the web service that is defined during application assembly.

Request generator (sender) binding:

Specifies the binding configuration that is used to send request messages to the request consumer.

Click **Edit custom** to configure the required and additional properties such as signing information, key information, token generators, key locators, and collection certificate stores.

The binding information for the request generator that is specified for the client must match the binding information for the request consumer that is specified for the server.

Response consumer (receiver) binding:

Specifies the binding configuration that is used to receive response messages from the response generator.

Click **Edit custom** to configure the required and additional properties such as signing information, key information, token consumers, key locators, collection certificate stores, and trust anchors.

The binding information for the response consumer that is specified for the client must match the binding information for the response generator that is specified for the server.

Request sender binding:

Specifies the binding configuration that is used to send request messages to the request receiver.

Click **Edit** to configure the additional properties for the request sender such as signing information, key information, encryption information, key locators, and the login binding.

The binding information for the request sender that is specified for the client must match the binding information for the request receiver that is specified for the server.

Response receiver binding:

Specifies the binding configuration that is used to receive response messages from the response sender.

Click **Edit** to configure the additional properties for the response receiver such as signing information, encryption information, trust anchors, collection certificate stores, and key locators.

The binding information for the response receiver that is specified for the client must match the binding information for the response sender that is specified for the server.

HTTP basic authentication:

Specifies the user name and password to use for this port with HTTP transport-level basic authentication. You can enable transport-level authentication security independently of message-level security.

Although the name of this field is HTTP basic authentication, you can use this field to specify the user name and password in conjunction with any transport method. This field is not specific to HTTP transport. For example, you can use this same field with Java Message Service (JMS).

Click **Edit** to configure the basic authentication ID and password for transport-level authentication.

HTTP SSL configuration:

Enables and configures transport-level Secure Sockets Layer (SSL) security for this port. You can enable transport-level SSL security independently of message-level security.

Click **Edit** to specify the settings for transport-level HTTP SSL configuration for this port.

Web services: Server security bindings collection:

Use this page to view a list of server-side binding configurations for Web Services Security.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, click **Web services: Server security bindings**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

Port:

Specifies the port in which messages are received from the request generator.

Web service:

Specifies the name of the web service that is defined during application assembly.

Request consumer (receiver) binding:

Specifies the binding configuration that is used to receive request messages from the request generator (sender) binding.

Click **Edit custom** to configure the required and additional information such as signing information, key information, token consumers, key locators, intermediate certificates in the collection certificate store, and trust anchors.

The binding information for the request consumer that is specified for the server must match the binding information for the request generator that is specified for the client.

Response generator (sender) binding:

Specifies the binding configuration that is used to send response messages to the response consumer.

Click **Edit custom** to configure the required and additional information such as signing information, key information, token generators, key locators, and intermediate certificates in the collection certificate store.

The binding information for the response generator that is specified for the server must match the binding information for the response consumer that is specified for the client.

Request receiver binding:

Specifies the binding configuration that is used to receive request messages from the request sender binding.

Click **Edit** to configure additional properties for the request receiver such as signing information, encryption information, trust anchors, collection certificate stores, key locators, trusted ID evaluators, and login mappings.

The binding information for the request receiver that is specified for the server must match the binding information for the request sender that is specified for the client.

Response sender binding:

Specifies the binding configuration that is used to send response messages to the response receiver.

Click **Edit** to configure additional properties for the response sender such as signing information, encryption information, and key locators.

The binding information for the response sender that is specified for the server must match the binding information for the response receiver that is specified for the client.

Configuring token consumers using JAX-RPC to protect message authenticity at the application level:

You can specify the token consumer on the application level. The token consumer information is used on the consumer side to incorporate the security token.

About this task

Complete the following steps to configure the token consumer on the application level.

Procedure

1. Locate the token consumer panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Modules, click **Manage modules** > ***URI_name***.
   c. Under Web Services Security Properties you can access the token consumer for the following bindings:
      - For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
      - For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   d. Under Required properties, click **Token consumer**.
   e. Click **New** to create a token consumer configuration, click **Delete** to delete an existing configuration, or click the name of an existing token consumer configuration to edit its settings. If you are creating a new configuration, enter a unique name in the **Token consumer name** field. For example, you might specify `con_signtcon`.
2. Specify a class name in the **Token consumer class name** field. The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to validate (authenticate) the security token on the consumer side.

   The token consumer class name for the request consumer and the response consumer must be similar to the token generator class name for the request generator and the response generator. For example, if your application requires a user name token consumer, you can specify the com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator class name on the Token generator panel for application level and the com.ibm.wsspi.wssecurity.token.UsernameTokenConsumer class name in this field.
3. Optional: Select a part reference in the **Part reference** field. The part reference indicates the name of the security token that is defined in the deployment descriptor. For example, if you receive a username token in your request message, you might want to reference the token in the username token consumer.

   Important: On the application level, if you do not specify a security token in your deployment descriptor, the **Part reference** field is not displayed. If you define a security token called `user_tcon` in your deployment descriptor, `user_tcon` is displayed as an option in the **Part reference** field.
4. Optional: In the certificate path section of the panel, select a certificate store type and indicate the trust anchor and certificate store name, if necessary. These options and fields are necessary when you specify com.ibm.wsspi.wssecurity.token.X509TokenConsumer as the token consumer class name. The names of the trust anchor and the collection certificate store are created in the certificate path under your token consumer.

   Restriction: The `com.ibm.wsspi.wssecurity.token.TokenConsumingComponent` interface is not used with JAX-WS web services. If you are using JAX-RPC web services, this interface is still valid.

   You can select one of the following options:

   None If you select this option, the certificate path is not specified.

   Trust any
   If you select this option, any certificate is trusted. When the received token is consumed, the Application Server does not validate the certificate path.

Dedicated signing information

If you select this option, you can select a trust anchor and a certificate store configuration. When you select the trust anchor or the certificate store of a trusted certificate, you must configure the trust anchor and the certificate store before setting the certificate path.

Trust anchor

A trust anchor specifies a list of key store configurations that contain trusted root certificates. These configurations are used to validate the certificate path of incoming X.509-formatted security tokens. Keystore objects within trust anchors contain trusted root certificates that are used by the CertPath API to validate the trustworthiness of a certificate chain. You must create the keystore file using the keytool utility. The keytool utility is available using the QShell Interpreter.

You can configure trust anchors for the application level by completing the following steps:

a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.

b. Under Related Items, click **EJB Modules** or **Web Modules** > ***URI_name***.

c. Access the token consumer from the following bindings:

- For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
- For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.

d. Under Additional properties, click **Trust anchors**.

Collection certificate store

A collection certificate store includes a list of untrusted, intermediary certificates and certificate revocation lists (CRLs). The collection certificate store is used to validate the certificate path of the incoming X.509-formatted security tokens. You can configure the collection certificate store for the application level by completing the following steps:

a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.

b. Under Related Items, click **EJB Modules** or **Web Modules** > ***URI_name***.

c. Access the token consumer from the following bindings:

- For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
- For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.

d. Under Additional properties, click **Collection certificate store**.

5. Optional: Specify a trusted ID evaluator. The trusted ID evaluator is used to determine whether to trust the received ID. You can select one of the following options:

None If you select this option, the trusted ID evaluator is not specified.

Existing evaluator definition

If you select this option, you can select one of the configured trusted ID evaluators. For example, you can select the **SampleTrustedIDEvaluator**, which is provided by WebSphere Application Server as an example.

Binding evaluator definition

If you select this option, you can configure a new trusted ID evaluator by specifying a trusted ID evaluator name and class name.

Trusted ID evaluator name

Specifies the name that is used by the application binding to refer to a trusted identity (ID) evaluator that is defined in the default bindings.

Trusted ID evaluator class name

Species the class name of the trusted ID evaluator. The specified trusted ID evaluator class name must implement the com.ibm.wsspi.wssecurity.id.TrustedIDEvaluator interface. The default TrustedIDEvaluator class is com.ibm.wsspi.wssecurity.id.TrustedIDEvaluatorImpl. When you use this default TrustedIDEvaluator class, you must specify the name and value properties for the default trusted ID evaluator to create the trusted ID list for evaluation. To specify the name and value properties, complete the following steps:

a. Under Additional properties, click **Properties** > **New**.

b. Specify the trusted ID evaluator name in the **Property** field. You must specify the name in the form, `trustedId_n` where `_n` is an integer from 0 to n.

c. Specify the trusted ID in the **Value** field.

For example:

```
property name="trustedId_0", value="CN=Bob,O=ACME,C=US"
property name="trustedId_1, value="user1"
```

If the distinguished name (DN) is used, the space is removed for comparison. See the programming model information in the documentation for an explanation of how to implement the com.ibm.wsspi.wssecurity.id.TrustedIDEvaluator interface. For more information, see Default implementations of the Web Services Security service provider programming interfaces.

Note: Define the trusted ID evaluator on the server level instead of the application level. To define the trusted ID evaluator on the server level, complete the following steps:

a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.

b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

c. Under Additional properties, click **Trusted ID evaluators**.

d. Click **New** to define a new trusted ID evaluator.

The trusted ID evaluator configuration is available only for the token consumer on the server-side application level.

6. Optional: Select the **Verify nonce** option. This option indicates whether to verify a nonce in the user name token if it is specified for the token consumer. Nonce is a unique, cryptographic number that is embedded in a message to help stop repeat, unauthorized attacks of user name tokens. The **Verify nonce** option is valid only when the incorporated token type is a user name token.
7. Optional: Select the **Verify timestamp** option. This option indicates whether to verify a time stamp in the user name token. The **Verify nonce** option is valid only when the incorporated token type is a user name token.
8. Specify the value type local name in the **Local name** field. This field specifies the local name of the value type for the consumed token. For a user name token and an X.509 certificate security token, WebSphere Application Server provides predefined local names for the value type.

Table 305. Uniform Resource Identifier (URI) and Local name combinations. The local name value indicates the type of consumed token.

URI	Local name	Description
A namespace URI is not applicable.	Specify `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3` as the local name value.	Specifies the name of an X.509 certificate token
A namespace URI is not applicable.	Specify `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1` as the local name value.	Specifies the name of the X.509 certificates in a PKI path
A namespace URI is not applicable.	Specify `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7` as the local name value.	Specifies a list of X509 certificates and certificate revocation lists (CRL) in a PKCS#7
Specify `http://www.ibm.com/websphere/appserver/tokentype/5.0.2` as the URI value.	Specify `LTPA` as the local name value.	Specifies a binary security token that contains an embedded Lightweight Third Party Authentication (LTPA) token.

9. Optional: Specify the value type URI in the **URI** field. This entry specifies the namespace URI of the value type for the consumed token.

   Remember: If you specify the token consumer for a username token or an X.509 certificate security token, you do not need to specify a value type URI.

   If you want to specify another token, you must specify both the local name and the URI. For example, if you have an implementation of your own custom token, you can specify `CustomToken` in the **Local name** field and `http://www.ibm.com/custom`
10. Click **OK** and **Save** to save the configuration.
11. Click the name of your token consumer configuration.
12. Under Additional properties, click **JAAS configuration**. The Java Authentication and Authorization Service (JAAS) configuration specifies the name of the JAAS configuration that is defined in the JAAS login panel. The JAAS configuration specifies how the token logs in on the consumer side.
13. Select a JAAS configuration from the **JAAS configuration name** field. The field specifies the name of the JAAS system of application login configuration. You can specify additional JAAS system and application configurations by clicking **Global security**. Under Authentication, click **Java Authentication and Authorization Service** and click either **Application logins** > **New** or **System logins** > **New**. Do not remove the predefined system or application login configurations. However, within these configurations, you can add module class names and specify the order in which WebSphere Application Server loads each module. WebSphere Application Server provides the following predefined JAAS configurations:

   ClientContainer
   This selection specifies the login configuration that is used by the client container applications. The configuration uses the CallbackHandler application programming interface (API) that is defined in the deployment descriptor for the client container. To modify this configuration, see the JAAS configuration panel for application logins.

   WSLogin
   This selection specifies whether all of the applications can use the WSLogin configuration to perform authentication for the security run time. To modify this configuration, see the JAAS configuration panel for application logins.

   DefaultPrincipalMapping
   This selection specifies the login configuration that is used by Java 2 Connectors (J2C) to map users to principals that are defined in the J2C authentication data entries. To modify this configuration, see the JAAS configuration panel for application logins.

system.LTPA_WEB
This selection processes login requests that are used by the web container such as servlets and JavaServer Pages (JSPs) files. To modify this configuration, see the JAAS configuration panel for system logins.

system.RMI_OUTBOUND
This selection processes RMI requests that are sent outbound to another server when the com.ibm.CSIOutboundPropagationEnabled property is `true`. This property is set in the CSIv2 authentication panel.

To access the panel, click **Security** > **Global security**. Under Authentication, click **RMI/IIOP security** > **CSIv2 Outbound authentication**. To set the `com.ibm.CSIOutboundPropagationEnabled` property, select **Security attribute propagation**. To modify this JAAS login configuration, see the JAAS - System logins panel.

system.wssecurity.X509BST
This selection verifies an X.509 binary security token (BST) by checking the validity of the certificate and the certificate path. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.PKCS7
This selection verifies an X.509 certificate within a PKCS7 object that might include a certificate chain, a certificate revocation list, or both. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.PkiPath
This section verifies an X.509 certificate with a public key infrastructure (PKI) path. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.UsernameToken
This selection verifies the basic authentication (user name and password) data. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.IDAssertionUsernameToken
This selection supports the use of identity assertion in Versions 6 and later applications to map a user name to a WebSphere Application Server credential principal. To modify this configuration, see the JAAS configuration panel for system logins.

system.WSS_INBOUND
This selection specifies the login configuration for inbound or consumer requests for security token propagation using Web Services Security. To modify this configuration, see the JAAS configuration panel for system logins.

system.WSS_OUTBOUND
This selection specifies the login configuration for outbound or generator requests for security token propagation using Web Services Security. To modify this configuration, see the JAAS configuration panel for system logins.

None With this selection, you do not specify a JAAS login configuration.

14. Click **OK** and then click **Save** to save the configuration.

Results

You have configured the token consumer for the application level.

What to do next

You must specify a similar token generator configuration for the application level.

Request consumer (receiver) binding configuration settings:

Use this page to specify the binding configuration for the request consumer.

To view this administrative console page, complete the following steps:

1. Click **Applications > Application Types > WebSphere enterprise applications***application_name*.
2. Click **Manage modules**.
3. Click the Uniform Resource Identifier (URI).
4. Under Web Services Security Properties, click **Web services: Server security bindings**.
5. Under Request consumer (receiver) binding, click **Edit custom**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

The security constraints or bindings are defined using the application assembly process before the application is installed.

This product provides assembly tools to assemble your application.

If the security constraints are defined in the application, you must either define the corresponding binding information or select the **Use defaults** option on this panel and use the default binding information for the server level. The default binding that is provided by this product is a sample. Do not use this sample in a production environment without modifying the configuration. The security constraints define what is signed or encrypted in the Web Services Security message. The bindings define how to enforce the requirements.

Digital signature security constraint (integrity)

The following table shows the required and optional binding information when the digital signature security constraint (integrity) is defined in the deployment descriptor.

Table 306. Binding information for digital signature security constraints. The binding information is used for signing or encrypting messages.

Information type	Required or optional
Signing information	Required
Key information	Required
Token consumer	Required
Key locators	Optional
Collection certificate store	Optional
Trust anchors	Optional
Properties	Optional

You can use the key locators, collection certificate stores, and trust anchors that are defined at the server level.

Encryption constraint (confidentiality)

The following table shows the required and optional binding information when the encryption constraint (confidentiality) is defined in the deployment descriptor.

Table 307. Binding information for encryption constraints. The binding information is used for signing or encrypting messages.

Information type	Required or optional
Encryption information	Required
Key information	Required
Token consumer	Required
Key locators	Optional

Table 307. Binding information for encryption constraints (continued). The binding information is used for signing or encrypting messages.

Information type	Required or optional
Collection certificate store	Optional
Trust anchors	Optional
Properties	Optional

You can use the key locators, collection certificate store, and trust anchors that are defined at the server level.

Security token constraint

The following table shows the required and optional binding information when the security token constraint is defined in the deployment descriptor.

Table 308. Binding information for security token constraints. The binding information is used for signing or encrypting messages.

Information type	Required or optional
Token consumer	Required
Collection certificate store	Optional
Trust anchors	Optional
Properties	Optional

You can use the collection certificate store and trust anchors that are defined at the server level.

Use defaults:

Select this option if you want to use the default binding information from the server level.

If you select this option, the application server checks for binding information on the server level.

Port:

Specifies the port in the web service that is defined during application assembly.

Web service:

Specifies the name of the web service that is defined during application assembly.

Response consumer (receiver) binding configuration settings:

Use this page to specify the binding configuration for the response consumer.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules**.
3. Click the Uniform Resource Identifier (URI).
4. Under Web Services Security Properties, click **Web services: Client security bindings**.
5. Under Response consumer (receiver) binding, click **Edit custom**.

Depending on your assigned security role when security is enabled, you might not have access to text entry fields or buttons to create or edit configuration data. Review the administrative roles documentation to learn more about the valid roles for the application server.

The security constraints or bindings are defined using the application assembly process before the application is installed.

This product provides assembly tools to assemble your application.

The default binding that is provided by this product is a sample. Do not use this sample in a production environment without modifying the configuration. The security constraints define what is signed or encrypted in the Web Services Security message. The bindings define how to enforce the requirements.

Digital signature security constraint (integrity)

The following table shows the required and optional binding information when the digital signature security constraint (integrity) is defined in the deployment descriptor.

Table 309. Binding information for digital signature security constraints. The binding information is used for validating digital signature.

Information type	Required or optional
Signing information	Required
Key information	Required
Token consumer	Optional
Key locators	Optional
Collection certificate store	Optional
Trust anchors	Optional
Properties	Optional

You can use the key locators, collection certificate stores, and trust anchors that are defined at the server level.

Encryption constraint (confidentiality)

The following table shows the required and optional binding information when the encryption constraint (confidentiality) is defined in the deployment descriptor.

Table 310. Binding information for encryption constraints. The binding information is used for decrypting messages.

Information type	Required or optional
Encryption information	Required
Key information	Required
Token consumer	Optional
Key locators	Optional
Collection certificate store	Optional
Trust anchors	Optional
Properties	Optional

You can use the key locators, collection certificate store, and trust anchors that are defined at the application or server level.

Security token constraint

The following table shows the required and optional binding information when the security token constraint is defined in the deployment descriptor.

Table 311. Binding information for security token constraints. The binding information is used for digital signature verification and for decrypting messages.

Information type	Required or optional
Token consumer	Required
Collection certificate store	Optional
Trust anchors	Optional

Table 311. Binding information for security token constraints (continued). The binding information is used for digital signature verification and for decrypting messages.

Information type	Required or optional
Properties	Optional

You can use the collection certificate store and trust anchors that are defined at the application or server level.

Use defaults:

Select this option if you want to use the default binding information from the server level.

Component:

Specifies the enterprise bean in an assembled Enterprise JavaBeans (EJB) module.

Port:

Specifies the port in the web service that is defined during application assembly.

Web service:

Specifies the name of the web service that is defined during application assembly.

JAAS configuration settings:

Use this page to specify the name of the Java Authentication and Authorization Service (JAAS) configuration that is defined in the JAAS login panel.

Complete the following steps to access this page on the server level:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Consumer Bindings, click **Token consumers** > ***token_consumer_name*** or click **New** to create a new token consumer.
4. Under Additional properties, click **JAAS configuration**.

Complete the following steps to access this page on the application level:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access the JAAS configuration settings for the following bindings:
   - For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Token consumers** > ***token_consumer_name*** or click **New** to create a new token consumer. Under Additional properties, click **JAAS configuration**.
   - For the Request consumer (receiver) binding, click **Web services: Server security binding**. Under Request consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Token consumers** > ***token_consumer_name*** or click **New** to create a new token consumer. Under Additional properties, click **JAAS configuration**.

Important: If you create a new token consumer, you must click **Apply** before you can proceed to the JAAS configuration.

JAAS configuration name:

Specifies the name of the JAAS system or application login configuration.

Do not remove the predefined system or application login configurations. However, within these configurations, you can add module class names and specify the order in which the application server loads each module.

Preconfigured system login configurations

The following predefined system login configurations are defined on the system logins panel, which is accessible by completing the following steps:

1. Click **Security** > **Global security**.
2. Expand Java Authentication and Authorization Service, click **System logins**.

system.wssecurity.IDAssertionUsernameToken
Enables a Version 6.x application to use identity assertion to map a user name to an application server credential principal.

system.wssecurity.IDAssertion
Enables an application to use identity assertion to map a user name to an application server credential principal.

system.wssecurity.Signature
Enables an application to map a distinguished name (DN) in a signed certificate to an application server credential principal.

system.LTPA_WEB
Processes login requests used by the web container such as servlets and JavaServer Pages (JSP) files.

system.RMI_OUTBOUND
Processes RMI requests that are sent outbound to another server when either the com.ibm.CSI.rmiOutboundLoginEnabled or the com.ibm.CSIOutboundPropagationEnabled properties are `true`. These properties are set in the Common Secure Interoperability Version 2 (CSIv2) authentication panel.

To access the panel, click **Security** > **Global security**. Epand RMI/IIOP security, click **CSIv2 Outbound authentication**. To set the `com.ibm.CSI.rmiOutboundLoginEnabled` property, select the **Custom outbound mapping** option. To set the `com.ibm.CSIOutboundPropagationEnabled` property, select the **Security attribute propagation** option.

system.wssecurity.509BST
Verifies an .509 binary security token (BST) by checking the validity of the certificate and the certificate path.

system.wssecurity.PKCS7
Verifies an .509 certificate with a certificate revocation list in a Public Key Cryptography Standards #7 (PKCS7) object.

system.wssecurity.PkiPath
Verifies an .509 certificate with a public key infrastructure (PKI) path.

system.wssecurity.UsernameToken
Verifies basic authentication (user name and password).

Application login configurations

The following predefined application login configurations are defined on the Application logins panel, which is accessible by completing the following steps:

1. Click **Security** > **Global security**.
2. Expand Java Authentication and Authorization Service, click **Application logins**.

ClientContainer
Specifies the login configuration that is used by the client container application. This application uses the CallbackHandler API that is defined in the deployment descriptor of the client container.

WSLogin
Specifies whether all applications can use the WSLogin configuration to perform authentication for the application server security run time.

DefaultPrincipalMapping
Specifies the login configuration that is used by Java 2 Connectors (J2C) to map users to principals that are defined in the J2C authentication data entries.

Configuring encryption using JAX-RPC to protect message confidentiality at the application level:

You can configure encryption information, used to specify how the generators (senders) encrypt outgoing messages, for the request generator (client side) and the response generator (server side) bindings at the application level.

Before you begin

Configure the key information that is referenced by the key information references in the encryption information panel.

About this task

This task provides the steps that are needed for configuring encryption information for the request generator (client side) and the response generator (server side) bindings at the application level. This encryption information is used to specify how the generators (senders) encrypt outgoing messages.

Complete the following steps to configure the encryption information for the request generator or response generator section of the bindings file on the application level:

Procedure

1. Locate the encryption information configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties, you can access the key information for the request generator and response generator bindings.
      - For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
      - For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
   d. Under Required properties, click **Encryption information**.
   e. Click **New** to create an encryption information configuration. Click **Delete** to delete an existing configuration or click the name of an existing encryption information configuration to edit its settings. If you are creating a new configuration, enter a name in the **Encryption information name** field. For example, you might specify `gen_encinfo`.

2. Select a data encryption algorithm from the **Data encryption algorithm** field. The selection specifies the algorithm that is used to encrypt parts of the message. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#tripledes-cbc
   - http://www.w3.org/2001/04/xmlenc#aes128-cbc
   - http://www.w3.org/2001/04/xmlenc#aes256-cbc

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.
   - http://www.w3.org/2001/04/xmlenc#aes192-cbc

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

     Restriction: Do not use the 192-bit key encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

     Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

   The data encryption algorithm that you select for the generator side must match the data encryption method that you select for the consumer side.
3. Select a key encryption algorithm from the **Key encryption algorithm** field. This selection specifies the algorithm that is used to encrypt keys. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.

     When running with Software Development Kit (SDK) Version 1.4, the list of supported key transport algorithms does not include this one. This algorithm appears in the list of supported key transport algorithms when running with SDK Version 1.5.

     Restriction: This algorithm is not supported when the WebSphere Application Server is running in Federal Information Processing Standard (FIPS) mode.

     By default, the RSA-OAEP algorithm uses the SHA1 message digest algorithm to compute a message digest as part of the encryption operation. Optionally, you can use the SHA256 or SHA512 message digest algorithm by specifying a key encryption algorithm property. For the property name, you can specify `com.ibm.wsspi.wssecurity.enc.rsaoaep.DigestMethod`. The property value is one of the following URIs of the digest method:
     - `http://www.w3.org/2001/04/xmlenc#sha256`
     - `http://www.w3.org/2001/04/xmlenc#sha512`

     By default, the RSA-OAEP algorithm uses a null string for the optional encoding octet string for the OAEPParams. You can provide an explicit encoding octet string by specifying a key encryption algorithm property. For the property name, you can specify `com.ibm.wsspi.wssecurity.enc.rsaoaep.OAEPparams`. The property value is the base 64-encoded value of the octet string.

     Important: You can set these digest method and OAEPParams properties on the generator side only. On the consumer side, these properties are read from the incoming SOAP message.
   - http://www.w3.org/2001/04/xmlenc#rsa-1_5
   - http://www.w3.org/2001/04/xmlenc#kw-tripledes

- http://www.w3.org/2001/04/xmlenc#kw-aes128
- http://www.w3.org/2001/04/xmlenc#kw-aes256

  To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.
- http://www.w3.org/2001/04/xmlenc#kw-aes192

  To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

  Restriction: Do not use the 192-bit key encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

The key encryption algorithm that you select for the generator side must match the key encryption method that you select for the consumer side.

4. Select an encryption key information reference from the Encryption key information menu. This selection is a reference to the encryption key that is used to encrypt parts of the message. To configure the key information, see "Configuring the key information using JAX-RPC for the generator binding on the application level" on page 3314.
5. Select a part reference from the **Part reference** field. This field specifies the name of the part reference for the generator binding element in the deployment descriptor.
6. Click **OK** and then click **Save** to save the configuration.

Results

The encryption information is configured for the generator binding at the application level.

What to do next

You must specify a similar encryption information configuration for the consumer.

Encryption information collection:

Use this page to specify the configuration for the encrypting and decrypting parameters. This configuration is used to encrypt and decrypt parts of the message, including the body and user name token.

To view the administrative console panel for the encryption information on the server level, complete the following steps:

1. Click **Servers > Server Types > WebSphere application servers >** *server_name*.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under either JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Encryption information**.

To view this administrative console page for the collection certificate store on the application level, complete the following steps:

1. Click **Applications > Application Types > WebSphere enterprise applications***application_name*.
2. Under Modules, click **Manage modules >** ***URI_name***.
3. Under Web Services Security Properties, you can access encryption information for the following bindings:

- For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**. Under Required properties, click **Encryption information**.
- For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Encryption information**.
- For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. Under Required properties, click **Encryption information**.
- For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Encryption information**.

4. Under Additional properties, you can access encryption information for the following bindings:
   - For the Request receiver, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**. Under Additional properties, click **Encryption information**.
   - For the Response receiver, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**. Under Additional properties, click **Encryption information**.

Encryption information name:

Specifies the name of the encryption information.

Key locator reference:

Specifies the name of the key locator configuration that retrieves the key for XML digital signature and XML encryption.

Key encryption algorithm: Specifies the algorithm that is used to encrypt and decrypt keys.

Data encryption algorithm: Specifies the algorithm that is used to encrypt and decrypt data.

Encryption information configuration settings: Message parts:

Use this page to configure the encryption and decryption parameters. You can use these parameters to encrypt and decrypt various parts of the message, including the body and the token.

To view the administrative console panel for the encryption information on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under either JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Encryption information**.
4. Click **New** to create a new encryption configuration or click the name of an existing encryption configuration.

To view this administrative console page for the encryption information on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Module update** > ***module_name***.

3. Under Web Services Security Properties, you can access encryption information for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**. Under Required properties, click **Encryption information**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Encryption information**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. Under Required properties, click **Encryption information**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Encryption information**.
4. Click either **New** to create a new encryption configuration or click the name of an existing encryption configuration.

Note: Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

Encryption information name:

Specifies the name for the encryption information.

Data type String

Data encryption algorithm:

Specifies the algorithm Uniform Resource Identifier (URI) of the data encryption method.

The following algorithms are supported:
- http://www.w3.org/2001/04/xmlenc#tripledes-cbc
- http://www.w3.org/2001/04/xmlenc#aes128-cbc
- http://www.w3.org/2001/04/xmlenc#aes256-cbc. To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html. For more information, see “Encryption information configuration settings: Methods” on page 3363.
- http://www.w3.org/2001/04/xmlenc#aes192-cbc. To use this algorithm, you must download the unrestricted JCE policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html. For more information, see the help topic Encryption information configuration settings: Methods.

  Restriction: Do not use the 192-bit data encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

By default, the Java Cryptography Extension (JCE) is shipped with restricted or limited strength ciphers. To use 192-bit and 256-bit Advanced Encryption Standard (AES) encryption algorithms, you must apply unlimited jurisdiction policy files. For more information, see the **Key encryption algorithm** field description.

Key locator reference:

Specifies the name of the key locator configuration that retrieves the key for XML digital signature and XML encryption.

The Key locator reference field is displayed for the request receiver and response receiver bindings.

You can configure these key locator reference options on the server level and the application level. The configurations that are listed in the field are a combination of the configurations on these two levels.

You can specify an encryption key configuration for the following bindings on the following levels:

Table 312. Encryption key binding configurations. Use these configurations to encrypt and decrypt various parts of a message.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Key locators**.
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Key locators**.
Request sender	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Request sender binding, click **Edit**. 4. Under Additional properties, click **Key locators**.
Request receiver	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**. 4. Under Additional properties, click **Key locators**.
Response sender	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Server security bindings**. Under Response sender binding, click **Edit**. 4. Under Additional properties, click **Key locators**.
Response receiver	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**. 4. Under Additional properties, click **Key locators**.

Key encryption algorithm:

Specifies the algorithm Uniform Resource Identifier (URI) of the key encryption method.

The following algorithms are provided by the application server:

- http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.

  When running with Software Development Kit (SDK) Version 1.4, the list of supported key transport algorithms does not include this one. This algorithm appears in the list of supported key transport algorithms when running with Software Development Kit (SDK) Version 1.5 or later.

  By default, the RSA-OAEP algorithm uses the SHA1 message digest algorithm to compute a message digest as part of the encryption operation. Optionally, you can use the SHA256 or SHA512 message digest algorithm by specifying a key encryption algorithm property. The property name is: `com.ibm.wsspi.wssecurity.enc.rsaoaep.DigestMethod`. The property value is one of the following URIs of the digest method:

  – `http://www.w3.org/2001/04/xmlenc#sha256`

  – `http://www.w3.org/2001/04/xmlenc#sha512`

  By default, the RSA-OAEP algorithm uses a null string for the optional encoding octet string for the OAEPParams. You can provide an explicit encoding octet string by specifying a key encryption algorithm property. For the property name, you can specify `com.ibm.wsspi.wssecurity.enc.rsaoaep.OAEPparams`. The property value is the base 64-encoded value of the octet string.

  Important: You can set these digest method and OAEPParams properties on the generator side only. On the consumer side, these properties are read from the incoming SOAP message.

- http://www.w3.org/2001/04/xmlenc#rsa-1_5
- http://www.w3.org/2001/04/xmlenc#kw-tripledes
- http://www.w3.org/2001/04/xmlenc#kw-aes128
- http://www.w3.org/2001/04/xmlenc#kw-aes192

  Restriction: Do not use the 192-bit data encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

- http://www.w3.org/2001/04/xmlenc#kw-aes256

IBM Software Development Kit Version 1.4:

For IBM i and IBM Software Development Kit Version 1.4, the tuning of Web Services Security is not required. The unrestricted jurisdiction policy files for IBM Software Development Kit Version 1.4 are automatically configured when the prerequisite software is installed.

- For IBM i (formerly known as IBM i V5R3) and IBM Software Development Kit Version 1.4, install product 5722AC3, Crypto Access Provider 128-bit.
- For IBM i 5.4 and IBM Software Development Kit Version 1.4, install product 5722SS1 Option 3, Extended Base Directory Support.

IBM Software Development Kit Version 1.5:

For IBM i 5.4 and IBM i (formerly known as IBM i V5R3) and IBM Software Development Kit 1.5, the restricted JCE jurisdiction policy files are configured, by default. You can download the unrestricted JCE jurisdiction policy files from the following website: IBM developer works: Security Information, Version 5

Note: If Java Platform, Standard Edition 6 (Java SE 6) 32-bit for IBM i is the enabled Java virtual machine (JVM) for your profile, substitute `/QOpenSys/QIBM/ProdData/JavaVM/jdk50/32bit/jre` for `/QIBM/ProdData/Java400/jdk15` as the path name in the following steps.

Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

To configure the unrestricted jurisdiction policy files for IBM i and the IBM Software Development Kit Version 1.5:

1. Make backup copies of these files:

```
/QIBM/ProdData/Java400/jdk15/lib/security/local_policy.jar
/QIBM/ProdData/Java400/jdk15/lib/security/US_export_policy.jar
```

2. Download the unrestricted policy files from IBM developer works: Security Information to the `/QIBM/ProdData/Java400/jdk15/lib/security` directory.
   a. Go to this website: http://www.ibm.com/developerworks/java/jdk/security/index.html
   b. Click **J2SE 5.0**.
   c. Scroll down and click **IBM SDK Policy files**. The Unrestricted JCE Policy files for the SDK website is displayed.
   d. Click **Sign in** and provide your IBM intranet ID and password.
   e. Select the appropriate unrestricted JCE policy files, and then click **Continue**.
   f. View the license agreement, and then click **I Agree**.
   g. Click **Download Now**.
3. Use the DSPAUT command to ensure *PUBLIC is granted*RX data authority but also ensure that no object authority is provided to both the `local_policy.jar` and the `US_export_policy.jar` files in the `/QIBM/ProdData/Java400/jdk15/lib/security` directory. For example:

```
DSPAUT OBJ('/qibm/proddata/java400/jdk15/lib/security/local_policy.jar')
```

4. Use the CHGAUT command to change authorization, if needed. For example:

```
CHGAUT OBJ('/qibm/proddata/java400/jdk15/lib/security/local_policy.jar')
USER(*PUBLIC) DTAAUT(*RX) OBJAUT(*NONE)
```

Custom algorithms on the server level

To specify custom algorithms on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Algorithm mappings**.
4. Click **New** to specify a new algorithm mapping or click the name of an existing configuration to modify its settings.
5. Under Additional properties, click **Algorithm URI**.
6. Click **New** to create a new algorithm URI. You must specify **Key encryption** in the **Algorithm type** field to have the configuration display in the **Key encryption algorithm** field on the Encryption information configuration settings panel.

Encryption key information:

Specifies the name of the key information reference that is used for encryption. This reference is resolved to the actual key by the specified key locator and defined in the key information.

You must specify either one or no encryption key configurations for the request generator and response generator bindings.

For the response consumer and the request consumer bindings, you can configure multiple encryption key references. To create a new encryption key reference, under Additional properties, click **Key information references**.

You can specify an encryption key configuration for the following bindings on the following levels:

Table 313. Encryption key binding configurations. Use these configurations to encrypt and decrypt various parts of a message.

Binding name	Server level or application level	Path
Default generator binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under JAX-RPC Default generator binding, click **Key information**.
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under JAX-RPC Default consumer binding, click **Key information**.
Request generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Under Web Services Security Properties, click **Web services: Client security bindings**. 4. Under Request generator (sender) binding, click **Edit custom**. 5. Under Required properties, click **Key information**.
Response generator (sender) binding	Application level	1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***. 2. Under Modules, click **Manage modules** > ***URI_name***. 3. Under Web Services Security Properties, click **Web services: Server security bindings**. 4. Under Response generator (sender) binding, click **Edit custom**. 5. Under Required properties, click **Key information**.

Part Reference:

Specifies the name of the <confidentiality> element for the generator binding or the <requiredConfidentiality> element for the consumer binding element in the deployment descriptor.

This field is available on the application level only.

Encryption information configuration settings: Methods:

Use this page to configure the encryption and decryption parameters for the signature method, digest method, and canonicalization method.

The specifications that are listed on this page for the signature method, digest method, and canonicalization method are located in the World Wide Web Consortium (W3C) document entitled, *XML Encryption Syntax and Processing: W3C Recommendation 10 Dec 2002*.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** and complete one of the following steps:

- Click **Manage modules** > ***URI_file_name*** > **Web Services: Client Security Bindings**. Under Request sender binding, click **Edit**. Under Web Services Security Properties, click **Encryption Information**.
- Under Modules, click **Manage modules** > ***URI_file_name*** > **Web Services: Server Security Bindings**. Under Response sender binding, click **Edit**. Under Web Services Security Properties, click **Encryption Information**.

2. Select **None** or **Dedicated encryption information**. The application server can have either one or no encryption configurations for the request sender and the response sender bindings. If you are not using encryption, select **None**. To configure encryption for either of these two bindings, select **Dedicated encryption information** and specify the configuration settings using the fields that are described in this topic.

Note: Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

Encryption information name:

Specifies the name for the encryption information.

Key locator reference:

Specifies the name that is used to reference the key locator.

You can configure these key locator reference options on the server level and the application level. The configurations that are listed in the field are a combination of the configurations on these two levels.

To configure the key locators on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Key locators**.

To configure the key locators on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access the key locators for the following bindings:
   - For the Request sender, click **Web services: Client security bindings**. Under Request sender binding, click **Edit**. Under Additional properties, click **Key locators**.
   - For the Request receiver, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**. Under Additional properties, click **Key locators**.
   - For the Response sender, click **Web services: Server security bindings**. Under Response sender binding, click **Edit**. Under Additional properties, click **Key locators**.
   - For the Response receiver, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**. Under Additional properties, click **Key locators**.

Encryption key name:

Specifies the name of the encryption key that is resolved to the actual key by the specified key locator.

Data type String

Key encryption algorithm:

Specifies the algorithm uniform resource identifier (URI) of the key encryption method.

The following algorithms are supported:

- http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.

  When running with IBM Software Development Kit (SDK) Version 1.4, the list of supported key transport algorithms does not include this one. This algorithm appears in the list of supported key transport algorithms when running with JDK 1.5 or later.

  By default, the RSA-OAEP algorithm uses the SHA1 message digest algorithm to compute a message digest as part of the encryption operation. Optionally, you can use the SHA256 or SHA512 message digest algorithm by specifying a key encryption algorithm property. The property name is: `com.ibm.wsspi.wssecurity.enc.rsaoaep.DigestMethod`. The property value is one of the following URIs of the digest method:
  - `http://www.w3.org/2001/04/xmlenc#sha256`
  - `http://www.w3.org/2001/04/xmlenc#sha512`

  By default, the RSA-OAEP algorithm uses a null string for the optional encoding octet string for the OAEPParams. You can provide an explicit encoding octet string by specifying a key encryption algorithm property. For the property name, you can specify `com.ibm.wsspi.wssecurity.enc.rsaoaep.OAEPparams`. The property value is the base 64-encoded value of the octet string.

  Important: You can set these digest method and OAEPParams properties on the generator side only. On the consumer side, these properties are read from the incoming SOAP message.
- http://www.w3.org/2001/04/xmlenc#rsa-1_5.
- http://www.w3.org/2001/04/xmlenc#kw-tripledes.
- http://www.w3.org/2001/04/xmlenc#kw-aes128.
- http://www.w3.org/2001/04/xmlenc#kw-aes192. To use the 192-bit key encryption algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file.

  Restriction: Do not use the 192-bit key encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).
- http://www.w3.org/2001/04/xmlenc#kw-aes256. To use the 256-bit key encryption algorithm, you must download the unrestricted JCE policy file.

Note: If an InvalidKeyException error occurs and you are using the 129xxx or 256xxx encryption algorithm, the unrestricted policy files might not exist in your configuration.

Java Cryptography Extension

By default, the Java Cryptography Extension (JCE) is shipped with restricted or limited strength ciphers. To use 192-bit and 256-bit Advanced Encryption Standard (AES) encryption algorithms, you must apply unlimited jurisdiction policy files.

Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

To download the policy files, complete one of the following sets of steps:

After completing these steps, two Java archive (JAR) files are placed in the Java virtual machine (JVM) `jre/lib/security/` directory.

IBM i and IBM Software Development Kit 1.4

For the IBM i and IBM Software Development Kit Version 1.4, the tuning of Web Services Security is not required. The unrestricted jurisdiction policy files for the IBM Software Development Kit Version 1.4 are automatically configured when the prerequisite software is installed.

For the IBM i 5.4 operating system and IBM Software Development Kit Version 1.4, the unrestricted jurisdiction policy files for the IBM Java Developer Kit 1.4 are automatically configured by installing product 5722SS1 Option 3, Extended Base Directory Support.

For IBM i (formerly known as IBM i V5R3) and IBM Software Development Kit Version 1.4, the unrestricted jurisdiction policy files for the IBM Software Development Kit Version 1.4 are automatically configured by installing product 5722AC3, Crypto Access Provider 128-bit.

IBM i and IBM Software Development Kit 1.5

For IBM i 5.4 and IBM i (formerly known as IBM i V5R3) and IBM Software Development Kit 1.5, the restricted JCE jurisdiction policy files are configured, by default. You can download the unrestricted JCE jurisdiction policy files from the following website: Security information: IBM J2SE 5 SDKs

To configure the unrestricted jurisdiction policy files for IBM i and the IBM Software Development Kit Version 1.5:

1. Make backup copies of these files:

```
/QIBM/ProdData/Java400/jdk15/lib/security/local_policy.jar
/QIBM/ProdData/Java400/jdk15/lib/security/US_export_policy.jar
```

2. Download the unrestricted policy files from IBM developer kit: Security information to the `/QIBM/ProdData/Java400/jdk15/lib/security` directory.
   a. Go to this website: IBM developer kit: Security information
   b. Click **J2SE 5.0**.
   c. Scroll down and click **IBM SDK Policy files**. The Unrestricted JCE Policy files for the SDK website is displayed.
   d. Click **Sign in** and provide your IBM intranet ID and password.
   e. Select the appropriate unrestricted JCE policy files, and then click **Continue**.
   f. View the license agreement, and then click **I Agree**.
   g. Click **Download Now**.
3. Use the DSPAUT command to ensure *PUBLIC is granted*RX data authority but also ensure that no object authority is provided to both the `local_policy.jar` and the `US_export_policy.jar` files in the `/QIBM/ProdData/Java400/jdk15/lib/security` directory. For example:

```
DSPAUT OBJ('/qibm/proddata/java400/jdk15/lib/security/local_policy.jar')
```

4. Use the CHGAUT command to change authorization, if needed. For example:

```
CHGAUT OBJ('/qibm/proddata/java400/jdk15/lib/security/local_policy.jar')
USER(*PUBLIC) DTAAUT(*RX) OBJAUT(*NONE)
```

Data encryption algorithm:

Specifies the algorithm Uniform Resource Identifiers (URI) of the data encryption method.

The following algorithms are supported:

- http://www.w3.org/2001/04/xmlenc#tripledes-cbc
- http://www.w3.org/2001/04/xmlenc#aes128-cbc
- http://www.w3.org/2001/04/xmlenc#aes192-cbc

Restriction: Do not use the 192-bit data encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

- http://www.w3.org/2001/04/xmlenc#aes256-cbc

By default, the JCE ships with restricted or limited strength ciphers. To use 192-bit and 256- bit AES encryption algorithms, you must apply unlimited jurisdiction policy files. For more information, see the Key encryption algorithm field description.

Configuring encryption to protect message confidentiality at the application level:

You can configure the encryption information for the request consumer (server side) and response consumer (client side) bindings at the application level.

Before you begin

Configure the key information that is referenced in the encryption information panel. For more information, see "Configuring the key information for the consumer binding on the application level" on page 3324.

About this task

This task provides the steps that are needed for configuring the encryption information for the request consumer (server side) and response consumer (client side) bindings at the application level. The encryption information on the consumer side is used for decrypting the encrypted message parts in the incoming SOAP message.

Complete the following steps to configure the encryption information for the request consumer or response consumer section of the bindings file on the application level:

Procedure

1. Locate the Encryption information configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties you can access the encryption information for the request consumer and response consumer bindings.
      - For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
      - For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   d. Under Required properties, click **Encryption information**.
   e. Click **New** to create an encryption information configuration, click **Delete** to delete an existing configuration, or click the name of an existing encryption information configuration to edit its settings. If you are creating a new configuration, enter a name in the **Encryption information name** field. For example, you might specify `cons_enc info`.
2. Select a data encryption algorithm from the **Data encryption algorithm** field. The data encryption algorithm is used for encrypting or decrypting parts of a SOAP message such as the SOAP body or the username token. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#tripledes-cbc
   - http://www.w3.org/2001/04/xmlenc#aes128-cbc
   - http://www.w3.org/2001/04/xmlenc#aes256-cbc

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

- http://www.w3.org/2001/04/xmlenc#aes192-cbc

  To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

  Restriction: Do not use the 192-bit key encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

  Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

The data encryption algorithm that you select for the consumer side must match the data encryption method that you select for the generator side.

3. Select a key encryption algorithm from the **Key encryption algorithm** field. The key encryption algorithm is used for encrypting the key that is used for encrypting the message parts within the SOAP message. Select **(none)** if the data encryption key, which is the key that is used for encrypting the message parts, is not encrypted. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.

     When running with Software Development Kit (SDK) Version 1.4, the list of supported key transport algorithms does not include this one. This algorithm appears in the list of supported key transport algorithms when running with SDK Version 1.5.

     Restriction: This algorithm is not supported when the WebSphere Application Server is running in Federal Information Processing Standard (FIPS) mode.
   - http://www.w3.org/2001/04/xmlenc#rsa-1_5
   - http://www.w3.org/2001/04/xmlenc#kw-tripledes
   - http://www.w3.org/2001/04/xmlenc#kw-aes128
   - http://www.w3.org/2001/04/xmlenc#kw-aes256

     To use the http://www.w3.org/2001/04/xmlenc#aes256-cbc algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.
   - http://www.w3.org/2001/04/xmlenc#kw-aes192

     To use the http://www.w3.org/2001/04/xmlenc#kw-aes192 algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

     Restriction: Do not use the 192-bit key encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

   The key encryption algorithm that you select for the consumer side must match the key encryption method that you select for the generator side.
4. Optional: Select a part reference in the **Part reference** field. The part reference specifies the name of the message part that is encrypted and is defined in the deployment descriptor. For example, you can encrypt the bodycontent message part in the deployment descriptor. The name of this Required Confidentiality part is `conf_con`. This message part is shown as an option in the **Part reference** field.
5. Under Additional properties, click **Key information references**.
6. Click **New** to create a key information configuration, click **Delete** to delete an existing configuration, or click the name of an existing key information configuration to edit its settings. If you are creating a new configuration, enter a name in the **Name** field. For example, you might specify `con_ekeyinfo`. This entry is the name of the <encryptionKeyInfo> element in the binding file.

7. Select a key information reference from the **Key information reference** field. This reference is the value of the keyinfoRef attribute of the <encryptionKeyInfo> element and it is the name of the <keyInfo> element that is referenced by this key information reference. Each key information reference entry generates an <encryptionKeyInfo> element under the <encryptionInfo> element in the binding configuration file. For example, if you enter `con_ekeyinfo` in the **Name** field and `dec_keyinfo` in the **Key information reference** field, the following <encryptionKeyInfo> element is generated in the binding file:

```
<encryptionKeyInfo xmi:id="EncryptionKeyInfo_1085092248843"
keyinfoRef="dec_keyinfo" name="con_ekeyinfo"/>
```

8. Click **OK** and then click **Save** to save the configuration.

Results

You have configured the encryption information for the consumer binding at the application level

What to do next

You must specify a similar encryption information configuration for the generator.

Configuring message-level security for JAX-RPC at the server or cell level

Specify the server-level or cell-level configuration.

Configuring the signing information using JAX-RPC for the generator binding on the server level:

You can configure the signing information for the client-side request generator and the server-side response generator bindings at the server level.

Before you begin

Note: For WebSphere Application Server version 6.x or earlier only, in the server-side extensions file (`ibm-webservices-ext.xmi`) and the client-side deployment descriptor extensions file (`ibm-webservicesclient-ext.xmi`), you must specify which parts of the message are signed. Also, you need to configure the key information that is referenced by the key information references on the Signing information panel within the administrative console.

About this task

This task explains the steps that are needed for you to configure the signing information for the client-side request generator and the server-side response generator bindings at the server level. WebSphere Application Server uses the signing information for the default generator to sign parts of the message that include the body, time stamp, and user name token if these bindings are not defined at the application level. The Application Server provides default values for bindings. However, an administrator must modify the defaults for a production environment.

Complete the following steps to configure the signing information for the generator sections of the bindings files on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default generator bindings, click **Signing information**.

3. Click **New** to create a signing information configuration, click **Delete** to delete an existing configuration, or click the name of an existing signing information configuration to edit the settings. If you are creating a new configuration, enter a unique name for the signing configuration in the Signing information name field. For example, you might specify `gen_signinfo`.
4. Select a signature method algorithm from the Signature method field. The algorithm that is specified for the default generator must match the algorithm that is specified for the default consumer. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2000/09/xmldsig#rsa-sha1
   - http://www.w3.org/2000/09/xmldsig#hmac-sha1
   - http://www.w3.org/2000/09/xmldsig#dsa-sha1

     Do not use this algorithm if you want the configured application to be compliant with the Basic Security Profile (BSP). Any ds:SignatureMethod/@Algorithm element in a SIGNATURE based on a symmetric key must have a value of http://www.w3.org/2000/09/xmldsig#rsa-sha1 or http://www.w3.org/2000/09/xmldsig#hmac-sha1.
5. Select a canonicalization method from the Canonicalization method field. The canonicalization algorithm that you specify for the generator must match the algorithm for the consumer. WebSphere Application Server supports the following pre-configured canonical XML and exclusive XML canonicalization algorithms:
   - http://www.w3.org/2001/10/xml-exc-c14n#
   - http://www.w3.org/2001/10/xml-exc-c14n#WithComments
   - http://www.w3.org/TR/2001/REC-xml-c14n-20010315
   - http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
6. Select a key information signature type from the **Key information signature type** field. The key information signature type determines how to digitally sign the key. WebSphere Application server supports the following signature types:

   None Specifies that the <KeyInfo> element is not signed.

   Keyinfo
   Specifies that the entire <KeyInfo> element is signed.

   Keyinfochildelements
   Specifies that the child elements of the <KeyInfo> element are signed.

   The key information signature type for the generator must match the signature type for the consumer. You might encounter the following situations:
   - If you do not specify one of the previous signature types, WebSphere Application Server uses `keyinfo`, by default.
   - If you select Keyinfo or Keyinfochildelements and you select http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform as the transform algorithm in a subsequent step, WebSphere Application Server also signs the referenced token.
7. Select a signing key information reference from the Signing key information field. This selection is a reference to the signing key that the Application Server uses to generate digital signatures. In the binding files, this information is specified within the <signingKeyInfo> tag. The key that is used for signing is specified by the key information element, which is defined at the same level as the signing information. For more information, see “Configuring the key information for the generator binding using JAX-RPC on the server level” on page 3375.
8. Click **OK** to save the configuration.
9. Click the name of the new signing information configuration. This configuration is the one that you specified in the previous steps.
10. Specify the part reference, digest algorithm, and transform algorithm. The part reference specifies which parts of the message to digitally sign.

a. Under Additional Properties, click **Part references** > **New** to create a new part reference, click **Part references** > **Delete** to delete an existing part reference, or click a part name to edit an existing part reference.

b. Specify a unique part name for the message part that needs signing. This message part is specified on both the server side and the client side. You must specify an identical part name for both the server side and the client side. For example, you might specify `reqint` for both the generator and the consumer.

   Important: You do not need to specify a value for the Part reference in the default bindings like you specify on the application level because the part reference on the application level points to a particular part of the message that is signed. Because the default bindings for the server level is applicable to all of the services that are defined on a particular server, you cannot specify this value.

c. Select a digest method algorithm in the **Digest method algorithm** field. The digest method algorithm that is specified in the binding files within the <DigestMethod> element is used in the <SigningInfo> element.

   WebSphere Application Server supports the following algorithms:

   - http://www.w3.org/2000/09/xmldsig#sha1
   - http://www.w3.org/2001/04/xmlenc#sha256
   - http://www.w3.org/2001/04/xmlenc#sha512

d. Click **OK** and **Save** to save the configuration.

e. Click the name of the new part reference configuration. This configuration is the one that you specified in the previous steps.

f. Under Additional properties, click **Transforms** > **New** to create a new transform, click **Transforms** > **Delete** to delete a transform, or click a transform name to edit an existing transform. If you create a new transform configuration, specify a unique name. For example, you might specify `reqint_body_transform1`.

g. Select a transform algorithm from the menu. The transform algorithm is specified within the <Transform> element. This algorithm element specifies the transform algorithm for the digital signature. WebSphere Application Server supports the following algorithms:

   - http://www.w3.org/2001/10/xml-exc-c14n#
   - http://www.w3.org/TR/1999/REC-xpath-19991116

     Restriction: Do not use this transform algorithm if you want your configured application to be compliant with the Basic Security Profile (BSP). Instead use http://www.w3.org/2002/06/xmldsig-filter2 to ensure compliance.

   - http://www.w3.org/2002/06/xmldsig-filter2
   - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform
   - http://www.w3.org/2002/07/decrypt#XML
   - http://www.w3.org/2000/09/xmldsig#enveloped-signature

   The transform algorithm that you select for the generator must match the transform algorithm that you select for the consumer.

   Important: If both of the following conditions are true, WebSphere Application Server signs the referenced token:

   - You previously selected the **Keyinfo** or the **Keyinfochildelements** option from the Key information signature type field on the signing information panel.
   - You select **http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform** as the transform algorithm.

11. Click **Apply**.

12. Click **Save** at the top of the panel to save your configuration.

Results

After completing these steps, you have configured the signing information for the generator on the server level.

What to do next

You must specify a similar signing information configuration for the consumer.

Configuring the signing information using JAX-RPC for the consumer binding on the server level:

You can configure the signing information for the client-side request generator and server-side response generator bindings at the server level.

Before you begin

Note: For WebSphere Application Server version 6.x or earlier only, in the server-side extensions file (`ibm-webservices-ext.xmi`) and the client-side deployment descriptor extensions file (`ibm-webservicesclient-ext.xmi`), you must specify which parts of the message are signed. Also, you need to configure the key information that is referenced by the key information references on the signing information panel within the administrative console.

About this task

This task explains the steps that are needed for you to configure the signing information for the client-side request generator and server-side response generator bindings at the server level. WebSphere Application Server uses the signing information for the default generator to sign parts of the message including the body, time stamp, and user name token, if these bindings are not defined at the application level. The Application Server provides default values for bindings. However, an administrator must modify the defaults for a production environment.

Complete the following steps to configure the signing information for the consumer sections of the bindings files on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default consumer bindings, click **Signing information**.
3. Click **New** to create a signing information configuration, click **Delete** to delete an existing configuration, or click the name of an existing signing information configuration to edit the settings. If you are creating a new configuration, enter a unique name for the signing configuration in the Signing information name field. For example, you might specify `gen_signinfo`.
4. Select a signature method algorithm from the Signature method field. The algorithm that is specified for the default consumer must match the algorithm that is specified for the default generator. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2000/09/xmldsig#rsa-sha1
   - http://www.w3.org/2000/09/xmldsig#hmac-sha1
   - http://www.w3.org/2000/09/xmldsig#dsa-sha1

Do not use this algorithm if you want the configured application to be compliant with the Basic Security Profile (BSP). Any ds:SignatureMethod/@Algorithm element in a SIGNATURE based on a symmetric key must have a value of http://www.w3.org/2000/09/xmldsig#rsa-sha1 or http://www.w3.org/2000/09/xmldsig#hmac-sha1.

5. Select a canonicalization method from the Canonicalization method field. The canonicalization algorithm that you specify for the generator must match the algorithm for the consumer. WebSphere Application Server supports the following pre-configured canonical XML and exclusive XML canonicalization algorithms:
   - http://www.w3.org/2001/10/xml-exc-c14n#
   - http://www.w3.org/2001/10/xml-exc-c14n#WithComments
   - http://www.w3.org/TR/2001/REC-xml-c14n-20010315
   - http://www.w3.org/TR/2001/REC-xml-c14n-20010315#WithComments
6. Select a key information signature type from the Key information signature type field. The key information signature type determines how to digitally sign the key. WebSphere Application Server supports the following signature types:

   None Specifies that the KeyInfo element is not signed.

   Keyinfo
   Specifies that the entire KeyInfo element is signed.

   Keyinfochildelements
   Specifies that the child elements of the KeyInfo element are signed.

   The key information signature type for the consumer must match the signature type for the generator. You might encounter the following situations:
   - If you do not specify one of the previous signature types, WebSphere Application Server uses keyinfo, by default.
   - If you select **Keyinfo** or **Keyinfochildelements** and you select **http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform** as the transform algorithm in a subsequent step, WebSphere Application Server also signs the referenced token.
7. Click **OK** to save the configuration.
8. Click the name of the new signing information configuration. This configuration is the one that you specified in the previous steps.
9. Specify the key information reference, part reference, digest algorithm, and transform algorithm.
   a. Under Additional properties, click **Key information references** > **New** to create a new reference, click **Key information references** > **Delete** to delete an existing reference, or click a reference name to edit an existing key information reference.
   b. Enter a name for the configuration in the Name field. For example, enter `con_skeyinfo`.
   c. Select a key information reference from the Key information reference field. The key Information reference points to the key that WebSphere Application Server uses for digital signing. In the binding files, the reference is specified within the <signingKeyInfo> element. The key that is used for signing is specified by the Key information element, which is defined at the same level as the signing information. For more information, see “Configuring the key information for the consumer binding on the application level” on page 3324.
   d. Click **OK** and **Save** to save the configuration.
   e. Under Additional Properties, click **Part references** > **New** to create a new part reference, click **Part references** > **Delete** to delete an existing part reference, or click a part name to edit an existing part reference. The part reference specifies which parts of the message to digitally sign. The part attribute refers to the name of the `<RequiredIntegrity>` element in the deployment descriptor when `<PartReference>` is specified for the digital signature. WebSphere Application Server enables you to specify multiple `<PartReference>` elements for the `<SigningInfo>` element. The `<PartReference>` element has two child elements: `<DigestMethod>` and `<Transform>`.
   f. Specify a unique part name for this part reference. For example, you might specify `reqint`.

Important: You do not need to specify a value for the Part reference field like you specify on the application level because the part reference on the application level points to a particular part of the message that is signed. Because the default bindings for the server level is applicable to all of the services that are defined on a particular server, you cannot specify this value.

g. Select a digest method algorithm in the **Digest method algorithm** field. The digest method algorithm specified within the `<DigestMethod>` element that is used in the `<SigningInfo>` element.

   WebSphere Application Server supports the following algorithms:

   - http://www.w3.org/2000/09/xmldsig#sha1
   - http://www.w3.org/2001/04/xmlenc#sha256
   - http://www.w3.org/2001/04/xmlenc#sha512

h. Click **OK** and **Save** to save the configuration.

i. Click the name of the new part reference configuration. This configuration is the one that you specified in the previous steps.

j. Under Additional properties, click **Transforms** > **New** to create a new transform, click **Transforms** > **Delete** to delete a transform, or click a transform name to edit an existing transform. If you create a new transform configuration, specify a unique name. For example, you might specify `reqint_body_transform1`.

k. Select a transform algorithm from the menu. The transform algorithm is specified within the <Transform> element. It specifies the transform algorithm for the signature. WebSphere Application Server supports the following algorithms:

   - http://www.w3.org/2001/10/xml-exc-c14n#
   - http://www.w3.org/TR/1999/REC-xpath-19991116

     Restriction: Do not use this transform algorithm if you want your configured application to be compliant with the Basic Security Profile (BSP). Instead use http://www.w3.org/2002/06/xmldsig-filter2 to ensure compliance.

   - http://www.w3.org/2002/06/xmldsig-filter2
   - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform
   - http://www.w3.org/2002/07/decrypt#XML
   - http://www.w3.org/2000/09/xmldsig#enveloped-signature

   The transform algorithm that you select for the consumer must match the transform algorithm that you select for the generator.

   Important: If both of the following conditions are true, WebSphere Application Server signs the referenced token:

   - You previously selected the Keyinfo or the Keyinfochildelements option from the Key information signature type field on the signing information panel.
   - You select http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#STR-Transform as the transform algorithm.

10. Click **OK**.
11. Click **Save** at the top of the panel to save your configuration.

Results

After completing these steps, you have configured the signing information for the consumer on the server level.

What to do next

You must specify a similar signing information configuration for the generator.

Configuring the key information for the generator binding using JAX-RPC on the server level:

Use the key information for the default generator to specify the key that is used by the signing or the encryption information configurations if these bindings are not defined at the application level.

About this task

The signing and encryption information configurations can share the same key information, which is why they are both defined on the same level. WebSphere Application Server provides default values for these bindings. However, an administrator must modify these values for a production environment.

Complete the following steps to configure the key information for the generator binding on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default generator bindings, click **Key information**.
3. Click **New** to create a key information configuration, click **Delete** to delete an existing configuration, or click the name of an existing key information configuration to edit the settings. If you are creating a new configuration, enter a unique name for the key configuration in the Key information name field. For example, you might specify `sig_keyinfo`.
4. Select a key information type from the Key information type field. WebSphere Application Server supports the following types of key information:

   Key identifier
   This key information type is used when two parties agree on how to create a key identifier. For example, a field of X.509 certificates can be used for the key identifier according to the X.509 profile.

   Key name
   This key information type is used when the sender and receiver agree on the name of the key.

   Security token reference
   This key information type is typically used when an X.509 certificate is used for digital signature.

   Embedded token
   This key information type is used to embed a security token in an embedded element.

   X509 issuer name and issuer serial
   This key information type specifies an X.509 certificate with its issuer name and serial number.

   Select **Security token reference** if you are using an X.509 certificate for the digital signature. In these steps, it is assumed that **Security token reference** is selected for this field.

   Important: This key information type must match the key information type that is specified for the consumer.

5. Select a key locator reference from the Key locator reference menu. In these steps, assume that the key locator reference is called `sig_klocator`. The key locator reference is the name of the key locator that is used to generate the key for digital signature. You must configure a key locator before you can select it in this field. For more information on configuring the key locator, see “Configuring the key locator using JAX-RPC on the server level” on page 3414.
6. Click **Get keys** to view a list of key name references. After you click **Get keys**, the key names that are defined in the <sig_klocator> element are shown in the key name reference menu. If you change the key locator reference, you must click **Get keys** again to display the list of key names that are associated with the new key locator.
7. Select a key name reference from the Key name reference menu. The key name reference specifies the name of the key that is used for generating the digital signature or for encryption. The Key name reference menu displays a list of key names that are defined for the selected key locator in the Key locator reference field. For example, select **signerkey**. It is assumed that signer key is a key name that is defined for the `sig_klocator` key locator.
8. Select a token reference from the Token reference field. The token reference refers to the name of a configured token generator. When a security token is required in the deployment descriptor, the token reference attribute is required. If you select **Security token reference** in the Key information type field, the token reference is required and you can specify an X.509 token generator. To specify an X.509 token generator, you must have an X.509 token generator configured. To configure an X.509 token generator, see “Configuring token generators using JAX-RPC to protect message authenticity at the server level” on page 3382. For the remaining steps, it is assumed that an X.509 token generator that is named gen_tcon is already configured.
9. Optional: Select an encoding method from the Encoding method field This field specifies the encoding format for the key identifier. The encoding method attribute is valid when you select **Key identifer** as the key information type. WebSphere Application Server supports the following encoding methods:
   - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary
   - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#HexBinary
10. Optional: Select a calculation method from the Calculation method field. The calculation method specifies the calculation algorithm that is used for the key identifier. This attribute is valid when you select **Key identifier** as the key information type. WebSphere Application Server supports the following calculation methods:
    - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#ITSHA1
    - http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#IT60SHA1
11. Optional: Specify a Uniform Resource Identifier (URI) of the value type for a security token from the Namespace URI field. The namespace URI is referenced by the key identifier. This attribute is valid when you select **Key identifier** as the key information type. When you specify the X.509 certificate token, you do not need to specify the namespace URI. If another token is specified, you must specify the namespace URI. For example, you can specify `http://www.ibm.com/websphere/appserver/tokentype/5.0.2` for the Lightweight Third Party Authentication (LTPA) token and `http://www.ibm.com/websphere/appserver/tokentype` for the LTPA_PROPAGATION token.
12. Optional: Specify the local name of the value type for a security token in the **Local name** field. The local name is referenced by the key identifier. This attribute is valid when you select **Key identifier** as the key information type. WebSphere Application Server supports the following local names:

    For an X.509 certificate token
    http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

    For X.509 certificates in a PKIPath
    http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1

    For a list of X.509 certificates and CRLs in a PKCS#7
    http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7

For LTPA
LTPA

For LTPA_PROPAGATION
LTPA_PROPAGATION

13. Click **OK** and **Save** to save the configuration.

Results

You have configured the key information for the generator binding at the server level.

What to do next

You must specify a similar key information configuration for the consumer.

Configuring the key information for the consumer binding using JAX-RPC on the server level:

The key information for the default consumer is used to specify the key for the signing or the encryption information configurations if these bindings are not defined at the application level.

About this task

The signing and encryption information configurations can share the same key information, which is why they are both defined on the same level. WebSphere Application Server provides default values for these bindings. However, an administrator must modify these values for a production environment.

Complete the following steps to configure the key information for the consumer binding on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default consumer bindings, click **Key information**.
3. Click **New** to create a key information configuration, click **Delete** to delete an existing configuration, or click the name of an existing key information configuration to edit the settings. If you are creating a new configuration, enter a unique name for the key configuration in the Key information name field. For example, you might specify `con_signkeyinfo`.
4. Select a key information type from the Key information type field. WebSphere Application Server supports the following types of key information:

   Key identifier
   This key information type is used when two parties agree on how to create a key identifier. For example, a field of X.509 certificates can be used for the key identifier according to the X.509 profile.

   Key name
   This key information type is used when the sender and receiver agree on the name of the key.

   Security token reference
   This key information type is typically used when an X.509 certificate is used for digital signature.

   Embedded token
   This key information type is used to embed a security token in an embedded element.

X509 issuer name and issuer serial
This key information type specifies an X.509 certificate with its issuer name and serial number.

Select **Security token reference** if you are using an X.509 certificate for the digital signature. In these steps, it is assumed that **Security token reference** is selected for this field.

Important: This key information type must match the key information type that is specified for the generator.

5. Select a key locator reference from the Key locator reference menu. In these steps, assume that the key locator reference is called `sig_klocator`. You must configure a key locator before you can select it in this field. For more information on configuring the key locator, see "Configuring the key locator using JAX-RPC on the server level" on page 3414.
6. Select a token reference from the Token reference field. The token reference refers to the name of a configured token consumer. When a security token is required in the deployment descriptor, the token reference attribute is required. If you select **Security token reference** in the Key information type field, the token reference is required and you can specify an X.509 token consumer. To specify an X.509 token consumer, you must have an X.509 token consumer configured. To configure an X.509 token consumer, see "Configuring token consumers using JAX-RPC to protect message authenticity at the server level" on page 3393.
7. Click **OK** and **Save** to save the configuration.

Results

You have configured the key information for the consumer binding at the server level.

What to do next

You must specify a similar key information configuration for the generator.

Configuring encryption using JAX-RPC to protect message confidentiality at the server or cell level:

You can configure the encryption information for the generator binding on the server or cell level.

About this task

The encryption information for the default generator specifies how to encrypt the information on the sender side if these bindings are not defined at the application level. WebSphere Application Server provides default values for the bindings. However, an administrator must modify the defaults for a production environment.

Complete the following steps to configure the encryption information for the generator binding on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default generator bindings, click **Encryption information**.
3. Click **New** to create an encryption information configuration, click **Delete** to delete an existing configuration, or click the name of an existing encryption information configuration to edit the settings.

If you are creating a new configuration, enter a unique name for the encryption configuration in the Encryption information name field. For example, you might specify `gen_encinfo`.

4. Select a data encryption algorithm from the Data encryption algorithm field. This algorithm is used to encrypt the data. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#tripledes-cbc
   - http://www.w3.org/2001/04/xmlenc#aes128-cbc
   - http://www.w3.org/2001/04/xmlenc#aes256-cbc

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.
   - http://www.w3.org/2001/04/xmlenc#aes192-cbc

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

     Restriction: Do not use this algorithm, the 192-bit key encryption algorithm, if you want your configured application to be in compliance with the Basic Security Profile (BSP).

   Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

   The data encryption algorithm that you select for the generator side must match the data encryption algorithm that you select for the consumer side.
5. Select a key encryption algorithm from the Key encryption algorithm field. This algorithm is used to encrypt the key. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.

     When running with JDK 1.4, the list of supported key transport algorithms will not include this one. This algorithm will appear in the list of supported key transport algorithms when running with JDK 1.5.

     Restriction: This algorithm is not supported when the WebSphere Application Server is running in Federal Information Processing Standard (FIPS) mode.

     By default, the RSA-OAEP algorithm uses the SHA1 message digest algorithm to compute a message digest as part of the encryption operation. Optionally, you can use the SHA256 or SHA512 message digest algorithm by specifying a key encryption algorithm property. The property name is: `com.ibm.wsspi.wssecurity.enc.rsaoaep.DigestMethod`. The property value is one of the following URIs of the digest method:
     - `http://www.w3.org/2001/04/xmlenc#sha256`
     - `http://www.w3.org/2001/04/xmlenc#sha512`

     By default, the RSA-OAEP algorithm uses a null string for the optional encoding octet string for the OAEPParams. You can provide an explicit encoding octet string by specifying a key encryption algorithm property. For the property name, you can specify `com.ibm.wsspi.wssecurity.enc.rsaoaep.OAEPparams`. The property value is the base 64-encoded value of the octet string.

     Important: You can set these digest method and OAEPParams properties on the generator side only. On the consumer side, these properties are read from the incoming SOAP message.
   - http://www.w3.org/2001/04/xmlenc#rsa-1_5
   - http://www.w3.org/2001/04/xmlenc#kw-tripledes

- http://www.w3.org/2001/04/xmlenc#kw-aes128
- http://www.w3.org/2001/04/xmlenc#kw-aes256

  To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.
- http://www.w3.org/2001/04/xmlenc#kw-aes192

  To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

  Restriction: Do not use this algorithm, the 192-bit key encryption algorithm, if you want your configured application to be in compliance with the Basic Security Profile (BSP).

If you select **None**, the key is not encrypted.

The key encryption algorithm that you select for the generator side must match the key encryption algorithm that you select for the consumer side.

6. Select a encryption key configuration from the Encryption key information field. This attribute specifies the name of the key that is used to encrypt the message. To configure the key information, see "Configuring the key information for the generator binding using JAX-RPC on the server level" on page 3375.
7. Click **OK** and then click **Save** to save the configuration.

Results

You have configured the encryption information for the generator binding at the server or cell level.

What to do next

You must specify a similar encryption information configuration for the consumer.

Configuring encryption to protect message confidentiality at the server level:

The encryption information for the default consumer specifies how to process the encryption information on the receiver side if these bindings are not defined at the application level. WebSphere Application Server provides default values for the bindings. However, an administrator must modify the defaults for a production environment.

About this task

Complete the following steps to configure the encryption information for the consumer binding on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default consumer bindings, click **Encryption information**.
3. Click **New** to create an encryption information configuration, click **Delete** to delete an existing configuration, or click the name of an existing encryption information configuration to edit the settings. If you are creating a new configuration, enter a unique name for the encryption configuration in the Encryption information name field. For example, you might specify `con_encinfo`.

4. Select a data encryption algorithm from the Data encryption algorithm field. This algorithm is used to encrypt the data. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#tripledes-cbc
   - http://www.w3.org/2001/04/xmlenc#aes128-cbc
   - http://www.w3.org/2001/04/xmlenc#aes256-cbc

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.
   - http://www.w3.org/2001/04/xmlenc#aes192-cbc

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

     Restriction: Do not use the 192-bit key encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

     Important: Your country of origin might have restrictions on the import, possession, use, or re-export to another country, of encryption software. Before downloading or using the unrestricted policy files, you must check the laws of your country, its regulations, and its policies concerning the import, possession, use, and re-export of encryption software, to determine if it is permitted.

   The data encryption algorithm that you select for the consumer side must match the data encryption algorithm that you select for the generator side.
5. Select a key encryption algorithm from the Key encryption algorithm field. This algorithm is used to encrypt the key. WebSphere Application Server supports the following pre-configured algorithms:
   - http://www.w3.org/2001/04/xmlenc#rsa-oaep-mgf1p.

     When running with Software Development Kit (SDK) Version 1.4, the list of supported key transport algorithms does not include this one. This algorithm appears in the list of supported key transport algorithms when running with SDK Version 1.5.

     Restriction: This algorithm is not supported when the WebSphere Application Server is running in Federal Information Processing Standard (FIPS) mode.
   - http://www.w3.org/2001/04/xmlenc#rsa-1_5
   - http://www.w3.org/2001/04/xmlenc#kw-tripledes
   - http://www.w3.org/2001/04/xmlenc#kw-aes128
   - http://www.w3.org/2001/04/xmlenc#kw-aes256

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.
   - http://www.w3.org/2001/04/xmlenc#kw-aes192

     To use this algorithm, you must download the unrestricted Java Cryptography Extension (JCE) policy file from the following website: http://www.ibm.com/developerworks/java/jdk/security/index.html.

     Restriction: Do not use the 192-bit key encryption algorithm if you want your configured application to be in compliance with the Basic Security Profile (BSP).

   If you select **None**, the key is not encrypted.

   The key encryption algorithm that you select for the consumer side must match the key encryption algorithm that you select for the generator side.
6. Under Additional properties, click **Key information references**.

7. Click **New** to create a key information configuration, click **Delete** to delete an existing configuration, or click the name of an existing key information configuration to edit the settings. If you are creating a new configuration, enter a unique name for the key information configuration in the name field. For example, you might specify `con_enckeyinfo`.
8. Select a key information reference from the Key information reference field. This selection refers to the name of the key information that is used for encryption. For more information, see "Configuring the key information for the consumer binding using JAX-RPC on the server level" on page 3377.
9. Click **OK** and **Save** to save the configuration.

Results

You have configured the encryption information for the consumer binding at the server level.

What to do next

You must specify a similar encryption information configuration for the generator.

Configuring token generators using JAX-RPC to protect message authenticity at the server level:

The token generator on the server level is used to specify the information for the token generator if these bindings are not defined at the application level. The signing information and the encryption information can share the token generator information, which is why they are all defined at the same level.

About this task

WebSphere Application Server provides default values for bindings. You must modify the defaults for a production environment.

Complete the following steps to configure the token generators on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default generator bindings, click **Token generators**.
3. Click **New** to create a token generator configuration, click **Delete** to delete an existing configuration, or click the name of an existing token generator configuration to edit its settings. If you are creating a new configuration, enter a unique name for the token generator configuration in the **Token generator name** field. For example, you might specify `sig_tgen`. This field specifies the name of the token generator element.
4. Specify a class name in the **Token generator class name** field. The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to create the security token on the generator side.

   Restriction: The com.ibm.wsspi.wssecurity.token.TokenGeneratorComponent interface is not used with JAX-WS web services. If you are using JAX-RPC web services, this interface is still valid.

   The token generator class name must be similar to the token consumer class name. For example, if your application requires an X.509 certificate token consumer, you can specify the com.ibm.wsspi.wssecurity.token.X509TokenConsumer class name on the Token consumer panel and

the com.ibm.wsspi.wssecurity.token.X509TokenGenerator class name in this field. WebSphere Application Server provides the following default token generator class implementations:

com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator
This implementation generates a username token.

com.ibm.wsspi.wssecurity.token.X509TokenGenerator
This implementation generates an X.509 certificate token.

com.ibm.wsspi.wssecurity.token.LTPATokenGenerator
This implementation generates a Lightweight Third Party Authentication (LTPA) token.

5. Select a certificate path option. The certificate path specifies the certificate revocation list (CRL), which is used for generating a security token that is wrapped in a PKCS#7 with a CRL. WebSphere Application Server provides the following certificate path options:

   None Select this option in case the CRL is not used for generating a security token. You must select this option when the token generator does not use the PKCS#7 token type.

   Dedicated signing information
   If the CRL is wrapped in a security token, select **Dedicated signing information** and select a collection certificate store name from the **Certificate store** field. The **Certificate store** field shows the names of collection certificate stores already defined.

6. Select the **Add nonce** option to include a nonce in the user name token for the token generator. Nonce is a unique cryptographic number that is embedded in a message to help stop repeat, unauthorized attacks of user name tokens. The **Add nonce** option is available if you specify a user name token for the token generator.
7. Select the **Add timestamp** option to include a time stamp in the user name token for the token generator.
8. Specify a value type local name in the **Local name** field. This entry specifies the local name of the value type for a security token that is referenced by the key identifier. This attribute is valid when **Key identifier** is selected as Key information type. To specify the Key information type, see “Configuring the key information for the generator binding using JAX-RPC on the server level” on page 3375. WebSphere Application Server provides the following predefined X.509 certificate token configurations:

   X.509 certificate token
   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

   X.509 certificates in a PKIPath
   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1

   A list of X.509 certificates and CRLs in a PKCS#7
   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7

   LTPA For LTPA, the value type local name is LTPA. If you enter `LTPA` for the local name, you must specify the http://www.ibm.com/websphere/appserver/tokentype/5.0.2 uniform resource identifier (URI) value in the **Value type URI** field as well.

   LTPA version 2
   For LTPA version 2, the value type local name is LTPAv2. If you enter LTPAv2 for the local name, you must specify the http://www.ibm.com/websphere/appserver/tokentype uniform resource identifier (URI) value in the **Value type URI** field as well.

   LTPA_PROPAGATION
   For LTPA token propagation, the value type local name is `LTPA_PROPAGATION`. If you enter `LTPA_PROPAGATION` for the local name, you must specify the `http://www.ibm.com/websphere/appserver/tokentype` URI value in the **Value type URI** field as well.

For example, when an X.509 certificate token is specified, you can use `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3` for the local name.

9. Specify the value type URI in the **URI** field. This entry specifies the namespace URI of the value type for a security token that is referenced by the key identifier. This attribute is valid when **Key identifier** is selected as Key information type on the Key information panel for the default generator. When the X.509 certificate token is specified, you do not need to specify the namespace URI. If another token is specified, you must specify the namespace URI of the value type.
10. Click **OK** and then **Save** to save the configuration.
11. Click the name of your token generator configuration.
12. Under Additional properties, click **Callback handler** to configure the callback handler properties. The callback handler specifies how to acquire the security token that is inserted in the Web Services Security header within the SOAP message. The token acquisition is a pluggable framework that leverages the Java Authentication and Authorization Service (JAAS) `javax.security.auth.callback.CallbackHandler` interface for acquiring the security token.
    a. Specify a callback handler class implementation in the **Callback handler class name** field. This attribute specifies the name of the Callback handler class implementation that is used to plug in a security token framework. The specified callback handler class must implement the `javax.security.auth.callback.CallbackHandler` class. WebSphere Application Server provides the following default callback handler implementations:

    com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler
    This callback handler uses a login prompt to gather the user name and password information. However, if you specify the user name and password on this panel, a prompt is not displayed and WebSphere Application Server returns the user name and password to the token generator. Use this implementation for a Java Platform, Enterprise Edition (Java EE) application client only.

    com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler
    This callback handler does not issue a prompt and returns the user name and password if it is specified in the basic authentication section of this panel. You can use this callback handler when the web service is acting as a client.

    com.ibm.wsspi.wssecurity.auth.callback.StdinPromptCallbackHandler
    This callback handler uses a standard-in prompt to gather the user name and password. However, if the user name and password is specified in the basic authentication section of this panel, WebSphere Application Server does not issue a prompt, but returns the user name and password to the token generator. Use this implementation for a Java Platform, Enterprise Edition (Java EE) application client only.

    com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler
    This callback handler is used to obtain the Lightweight Third Party Authentication (LTPA) security token from the Run As invocation Subject. This token is inserted in the Web Services Security header within the SOAP message as a binary security token. However, if the user name and password are specified in the basic authentication section of this panel, WebSphere Application Server authenticates the user name and password to obtain the LTPA security token. It obtains the security token this way rather than obtaining it from the Run As Subject. Use this callback handler only when the web service is acting as a client on the application server. It is recommended that you do not use this callback handler on a Java EE application client.

    com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler
    This callback handler is used to create the X.509 certificate that is inserted in the Web Services Security header within the SOAP message as a binary security token. A keystore file and a key definition are required for this callback handler.

    com.ibm.wsspi.wssecurity.auth.callback.PKCS7CallbackHandler
    This callback handler is used to create X.509 certificates that are encoded with the PKCS#7 format. The certificate is inserted in the Web Services Security header in the

SOAP message as a binary security token. A keystore file is required for this callback handler. You must specify a certificate revocation list (CRL) in the collection certificate store. The CRL is encoded with the X.509 certificate in the PKCS#7 format. For more information on configuring the collection certificate store, see “Configuring the collection certificate on the server level” on page 3433.

com.ibm.wsspi.wssecurity.auth.callback.PkiPathCallbackHandler
This callback handler is used to create X.509 certificates that are encoded with the PkiPath format. The certificate is inserted in the Web Services Security header within the SOAP message as a binary security token. A keystore file is required for this callback handler. A CRL is not supported by the callback handler; therefore, the collection certificate store is not required or used.

For an X.509 certificate token, you might specify the `com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler` implementation.

b. Optional: Select the **Use identity assertion** option. Select this option if you have identity assertion that is defined in the IBM extended deployment descriptor. This option indicates that only the identity of the initial sender is required and inserted into the Web Services Security header within the SOAP message. For example, WebSphere Application Server sends only the user name of the original caller for a user name token generator. For an X.509 token generator, the application server sends the original signer certification only.

c. Optional: Select the **Use RunAs identity** option. Select this option if the following conditions are true:
   - You have identity assertion defined in the IBM extended deployment descriptor.
   - You want to use the Run As identity instead of the initial caller identity for identity assertion for a downstream call.

d. Optional: Specify a basic authentication user ID and password in the **User ID** and **Password** fields. This entry specifies the user name and password that is passed to the constructors of the callback handler implementation. The basic authentication user ID and password are used if you specify one of the following default callback handler implementations that are provided by WebSphere Application Server:
   - com.ibm.wsspi.wssecurity.auth.callback.GUIPromptCallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.LTPATokenCallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.StdinPromptCallbackHandler

e. Optional: Specify a keystore password and path. The keystore and its related information are necessary when the key or certificate is used for generating a token. For example, the keystore information is required if you select one of the following default callback handler implementations that are provided by WebSphere Application Server:
   - com.ibm.wsspi.wssecurity.auth.callback.PKCS7CallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.PkiPathCallbackHandler
   - com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler

   The keystore files contain public and private keys, root certificate authority (CA) certificates, intermediate CA certificates, and so on. Keys that are retrieved from the keystore file are used to sign and validate or encrypt and decrypt messages or message parts. To retrieve a key from a keystore file, you must specify the keystore password, the keystore path, and the keystore type.

13. Select a keystore type from the **Type** field. WebSphere Application Server provides the following options:

JKS Use this option if you are not using Java Cryptography Extensions (JCE) and if your keystore file uses the Java Keystore (JKS) format.

JCEKS
Use this option if you are using Java Cryptography Extensions.

PKCS11KS (PKCS11)
Use this format if your keystore file uses the PKCS#11 file format. Key store files using this format might contain RSA keys on cryptographic hardware or might encrypt the keys that use cryptographic hardware to ensure protection.

PKCS12KS (PKCS12)
Use this option if your keystore file uses the PKCS#12 file format.

14. Click **OK** and then **Save** to save the configuration.
15. Click the name of your token generator configuration.
16. Under Additional properties, click **Callback handler** > **Keys**.
17. Click **New** to create a key configuration, click **Delete** to delete an existing configuration, or click the name of an existing key configuration to edit its settings. If you are creating a new configuration, enter a unique name for the key configuration in the **Key name** field. This name refers to the name of the key object that is stored within the keystore file.
18. Specify an alias for the key object in the **Key alias** field. Use the alias when the key locator searches for the key objects in the keystore.
19. Specify the password that is associated with the key in the **Key password** field.
20. Click **OK** and **Save** to save the configuration.

Results

You have configured the token generators at the server level.

What to do next

You must specify a similar token consumer configuration.

Token generator collection:

Use this page to view the token generators. The information is used on the generator side only to generate the security token.

To view this administrative console page for the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings, click **Token generators**.

Token generator name:

Specifies the name of the token generator configuration.

For example, the default X509 token generator names are either `gen_enctgen` for encrypting or `gen_signtgen` for signing. Or a custom token generator name might be `sig_tgen` for signing.

Token generator class name:

Specifies the name of the token generator implementation class.

This class must implement the `com.ibm.wsspi.wssecurity.token.TokenGeneratorComponent` interface.

Token generator class name:

Specifies the name of the token generator implementation class.

The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to create the security token on the generator side.

Token generator configuration settings:

Use this page to specify the information for the token generator. The information is used at the generator side only to generate the security token.

To view this administrative console page for the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Generator Bindings, click **Token generators** > ***token_generator_name*** or click **New** to create a new token generator.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Additional properties, you can access the token generator information for the following bindings:
   - For the Request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
4. Click **New** to create a new token generator or click the name of an existing token generator name to specify its settings.

To view this administrative console page for the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, click **Web services: Client security bindings**.
4. Under Request generator (sender) binding, click **Edit custom**.
5. Under Additional properties, click **Token generators** > **New**.

Before specifying additional properties, specify a value in the **Token generator name** and the **Token generator class name** fields.

Token generator name:

Specifies the name of the token generator configuration.

For example, the default X509 token generator names are either `gen_enctgen` for encrypting or `gen_signtgen` for signing. Or, a custom token generator name might be `sig_tgen` for signing.

Token generator class name:

Specifies the name of the token generator implementation class.

This class must implement the `com.ibm.wsspi.wssecurity.token.TokenGeneratorComponent` interface.

Token generator class name:

Specifies the name of the token generator implementation class.

Certificate path:

Specifies the certificate revocation list (CRL) that is used for generating a security token wrapped in a PKCS#7 token type with CRL.

When the token generator is not for a PKCS#7 token type, you must select **None**. When the token generator is for the PKCS#7 token type and you want to package CRL in the security token, select **Dedicated signing information** and specify the CRL for the collection certificate store.

You can specify a certificate store configuration for the following bindings on the following levels:

Table 314. Certificate path binding settings. The certificate is used for signing messages.

Binding name	Server level or application level	Path
Default generator bindings	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Collection certificate store**.

Using the collection certificate store, you can configure a related certificate revocation list by clicking **Certificate revocation list** under Additional properties.

Add nonce:

Indicates whether nonce is included in the user name token for the token generator. *Nonce* is a unique cryptographic number that is embedded in a message to help stop repeat, unauthorized attacks of user name tokens.

On the application level, if you select the **Add nonce** option, you can specify the following properties under Additional properties:

Table 315. Additional nonce properties. Nonce is used to add additional security to a message.

Property name	Default value	Explanation
`com.ibm.ws.wssecurity.config.token. BasicAuth.Nonce.cacheTimeout`	600 seconds	Specifies the timeout value, in seconds, for the nonce value that is cached on the server.
`com.ibm.ws.wssecurity.config.token. BasicAuth.Nonce.clockSkew`	0 seconds	Specifies the time, in seconds, before the nonce time stamp expires.
`com.ibm.ws.wssecurity.config.token. BasicAuth.Nonce.maxAge`	300 seconds	Specifies the clock skew value, in seconds, to consider when the application server checks the timeliness of the message.

These properties are available on the administrative console at the cell and server level. However, on the application level, you can configure the properties under Additional properties.

This option is displayed on the cell, server, and application levels. This option is valid only when the generated token type is a user name token.

Add timestamp:

Specifies whether to insert the time stamp into the user name token.

This option is displayed on the cell, server, and application levels. This option is valid only when the generated token type is a user name token.

Value type local name:

Specifies the local name of the value type for the generated token.

For a user name token and an X.509 certificate security token, this product provides predefined value types. When you specify the following local names, you do not need to specify the Uniform Resource Identifier (URI) of value type.

Username token
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

X509 certificate token
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

X509 certificates in a PKIPath
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1

A list of X509 certificates and CRLs in a PKCS#7
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7

Lightweight Third Party Authentication (LTPA)

LTPA_PROPAGATION

Important: For LTPA, the value type local name is `LTPA`. If you enter `LTPA` for the local name, you must specify the `http://www.ibm.com/websphere/appserver/tokentype/5.0.2` URI value in the Value type URI field as well. For LTPA token propagation, the value type local name is `LTPA_PROPAGATION`. If you enter `LTPA_PROPAGATION` for the local name, you must specify the `http://www.ibm.com/websphere/appserver/tokentype` URI value in the Value type URI field as well. For the other predefined value types (Username token, X509 certificate token, X509 certificates in a PKIPath, and a list of X509 certificates and CRLs in a PKCS#7), the value for the local name field begins with `http://`. For example, if you are specifying the user name token for the value type, enter `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken` in the Value type local name field and then you do not need to enter a value in the Value type URI field.

When you specify a custom value type for custom tokens, you can specify the local name and the URI of the quality name (QName) of the value type. For example, you might specify `Custom` for the local name and `http://www.ibm.com/custom` for the URI.

Value type URI:

Specifies the namespace URI of the value type for the generated token.

When you specify the token generator for the user name token or the X.509 certificate security token, you do not need to specify this option. If you want to specify another token, specify the URI of the QName of the value type.

The application server provides the following predefined value type URIs:

- For the LTPA token: http://www.ibm.com/websphere/appserver/tokentype/5.0.2
- For the LTPA token propagation: http://www.ibm.com/websphere/appserver/tokentype

Algorithm URI collection:

Use this page to view a list of uniform resource identifier (URI) algorithms for XML digital signature or XML encryption that are mapped to an algorithm factory engine class. With algorithm mappings, service providers can use other cryptographic algorithms for digest value calculation, digital signature signing and verification, data encryption and decryption, and key encryption and decryption.

To view administrative console page on the server level, complete the following steps:

1. Click **Servers > Server Types > WebSphere application servers >** ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Algorithm mappings**.
4. Click on an algorithm mapping name.
5. Under Additional properties, click **Algorithm URI**.

Algorithm URI:

Specifies the algorithm uniform resource identifier (URI) for the specified algorithm type.

Algorithm type:

Specifies the algorithm type.

Algorithm URI configuration settings:

Use this page to specify the algorithm uniform resource identifier (URI) and its usage type.

This product supports the following algorithm URI types:

Message digest
Specifies the algorithm URI that is used for digest value calculation.

Signature
Specifies the algorithm URI that is used for digital signature, including both signature and signing verification.

Data encryption
Specifies the algorithm URI that is used for both encrypting and decrypting data.

Key encryption
Specifies the algorithm URI that is used for encrypting and decrypting the encryption key.

If the URI is used for multiple usage types, then you must define a mapping of the URI to each usage type.

To view this administrative console page on the server level, complete the following steps:

1. Click **Servers > Server Types > WebSphere application servers >** *server_name*.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Algorithm mappings**.

   Note: The Algorithm mappings feature is not supported when the **Use the Federal Information Processing Standard (FIPS)** option has been selected on the SSL certificate and key

management panel of the administrative console. When this option is selected, the **New** button in the Algorithm mappings panel is not available.

4. Click **New**.
5. Under Additional properties, click **Algorithm URI** > ***algorithm_URI_name***.

To view the administrative console page on the cell level:

1. Click **Security > JAX-WS and JAX-RPC security runtime**, or **Services > JAX-WS and JAX-RPC security runtime**.
2. Under Additional properties, click **Algorithm mappings**.
3. Click **New**.
4. Under Additional properties, click **Algorithm URI** > ***algorithm_URI_name***.

Algorithm URI:

Specifies the algorithm uniform resource identifier (URI) for the specified algorithm type.

The algorithm URI that is defined on this page is available to the various binding configurations. For example, if you specify an algorithm URI and select **Signature** from the Algorithm type field, the URI displays in the Signature method field on the signing information panel.

Algorithm type:

Specifies the type of algorithm that is specified in the Algorithm URI field.

The following types of algorithms are supported by this product. The following list shows where configurations that are specified on this panel are displayed for a binding configuration:

Table 316. Algorithm types. The algorithm types in the table are supported by the product.

Algorithm type	Explanation	Location of the configuration
Signature	This algorithm type is used for digital signatures.	This configuration displays in the Signature method field on the Signing information panel. For information on how to access the Signing information panel, see the help topic Signing information configuration settings.
Digest value calculation (message digest)	This algorithm type is used for calculating the digest value.	This configuration displays in the Digest method algorithm field on the Part references panel. For information on how to access the Part references panel, see the help topic Part reference configuration settings.
Data encryption	This algorithm type is used for encrypting data.	This configuration displays in the Data encryption algorithm field on the Encryption information panel. For information on how to access the Encryption information panel, see the help topic Encryption information configuration settings: Message parts.
Key encryption	This algorithm type is used for encrypting the key that is used for data encryption.	This configuration displays in the Key encryption algorithm field on the Encryption information panel. For information on how to access the Encryption information panel, see the help topic Encryption information configuration settings: Message parts.

The actual implementation of the algorithm is done in the implementation class for the engine factory.

Algorithm mapping collection:

You can view a list of custom uniform resource identifier (URI) algorithms for digest value calculation, signature, key encryption, and data encryption. The application server maps these algorithms to an implementation of the algorithm factory engine interface. With algorithm mappings, service providers can extend the cryptographic algorithms for XML digital signature and XML encryption.

To view this administrative console page on the cell level, complete the following steps:

1. Click **Security > JAX-WS and JAX-RPC security runtime**.
2. Under Additional properties, click **Algorithm mappings**.

To view this administrative console page on the server level, complete the following steps:

1. Click **Servers > Server Types > WebSphere application servers >** ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Algorithm mappings**.

Algorithm factory engine class:

Specifies the custom class that implements the engine factory implementation class for the algorithm factory engine.

The implementation class for the engine factory implements the cryptographic functions of the defined uniform resource identifier (URI).

Note: The Algorithm mappings feature is not supported when the **Use the Federal Information Processing Standard (FIPS) algorithms** option has been selected on the Global security panel of the administrative console. When this option is selected, the **New** button in the Algorithm mappings panel is not available.

Algorithm mapping configuration settings:

Use this page to view a list of custom uniform resource identifier (URI) algorithms for digest value calculation, signature, key encryption, and data encryption. The application server maps these algorithms to an implementation of the algorithm factory engine interface. With algorithm mappings, service providers can extend the cryptographic algorithms for XML digital signature and XML encryption.

To view this administrative console page on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Algorithm mappings*****algorithm_factory_engine_class_name***.

   Note: The Algorithm mappings feature is not supported when the **Use the Federal Information Processing Standard (FIPS)** option has been selected on the SSL certificate and key management panel of the administrative console. When this option is selected, the **New** button in the Algorithm mappings panel is not available.
4. Click **New**.

To view this administrative console page on the cell level:

1. Click **Security** > **JAX-WS and JAX-RPC security runtime**, or **Services** > **JAX-WS and JAX-RPC security runtime**.
2. Under Additional properties, click **Algorithm mappings** > ***algorithm_factory_engine_class_name***.
3. Click **New**.

Algorithm factory engine class:

Specifies the custom class that implements the engine factory interface.

To use this algorithm mapping feature, you must specify a custom algorithm class in the Algorithm factory engine class field for digital signature, data encryption, digest value calculation, and key encryption. The

algorithm factory engine provides a plug-in point for service providers to provide their implementation for digest value calculation, digital signature, key encryption, and data encryption that is based on a specified algorithm uniform resource identifier (URI). By clicking **Algorithm URI** under Additional properties, you can specify the algorithm URI and its usage type. This product supports the following algorithm types:

Message digest
Specifies the algorithm URI that is used for digest value calculation.

Signature
Specifies the algorithm URI that is used for digital signatures including both signing and signature verification.

Data encryption
Specifies the algorithm URI that is used for both encrypting and decrypting data.

Key encryption
Specifies the algorithm URI that is used for both encrypting and decrypting the encryption key.

If the URI is used for multiple usage types, then you must define a mapping of the URI to each usage type. The actual implementation of the algorithm is provided by the custom class that implements the engine factory interface. For more information, refer to the information center documentation on how to implement a factory class.

By clicking **Properties** under Additional properties, you can specify name-value pair properties for the factory class.

Configuring token consumers using JAX-RPC to protect message authenticity at the server level:

The token consumer on the server level is used to specify the information that is needed to process the security token if it is not defined at the application level.

About this task

WebSphere Application Server provides default values for bindings. You must modify the defaults for a production environment.

Complete the following steps to configure the token consumers on the server level.

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Default consumer bindings, click **Token consumers**.
3. Click **New** to create a token consumer configuration, click **Delete** to delete an existing configuration, or click the name of an existing token consumer configuration to edit its settings. If you are creating a new configuration, enter a unique name for the token consumer configuration in the **Token consumer name** field. For example, you might specify `sig_tcon`. This field specifies the name of the token consumer element.
4. Specify a class name in the Token consumer class name field. The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to validate (authenticate) the security token on the consumer side.

Restriction: The com.ibm.wsspi.wssecurity.token.TokenConsumingComponent interface is not used with JAX-WS web services. If you are using JAX-RPC web services, this interface is still valid.

The token consumer class name must be similar to the token generator class name.

For example, if your application requires an X.509 certificate token consumer, you can specify the com.ibm.wsspi.wssecurity.token.X509TokenGenerator class name on the Token generator panel and the com.ibm.wsspi.wssecurity.token.X509TokenConsumer class name in this field. WebSphere Application Server provides the following default token consumer class implementations:

com.ibm.wsspi.wssecurity.token.UsernameTokenConsumer
This implementation integrates a user name token.

com.ibm.wsspi.wssecurity.token.X509TokenConsumer
This implementation integrates an X.509 certificate token.

com.ibm.wsspi.wssecurity.token.LTPATokenConsumer
This implementation integrates a Lightweight Third Party Authentication (LTPA) token.

com.ibm.wsspi.wssecurity.token.IDAssertionUsernameTokenConsumer
This implementation integrates an IDAssertionUsername token.

A corresponding token generator class does not exist for this implementation.

5. Select a certificate path option. The certificate path specifies the certificate revocation list (CRL) that is used for generating a security token wrapped in a PKCS#7 with a CRL. WebSphere Application Server provides the following certificate path options:

   None If you select this option, the certificate path is not specified.

   Trust any
   If you select this option, any certificate is trusted. When the received token is consumed, the certificate path validation is not processed.

   Dedicated signing information
   If you select this option, you can specify a trust anchor and a certificate store. When you select the trust anchor or the certificate store of a trusted certificate, you must configure the collection certificate store before setting the certificate path. To define a collection certificate store on the server level, see “Configuring the collection certificate on the server level” on page 3433.

   a. Select a trust anchor in the Trust anchor field. WebSphere Application Server provides two sample trust anchors. However, it is recommended that you configure your own trust anchors for a production environment. For information on configuring a trust anchor, see “Configuring trust anchors on the server level” on page 3422.
   b. Select a collection certificate store in the Certificate store field. WebSphere Application Server provides a sample collection certificate store. If you select **None**, the collection certificate store is not specified. For information on specifying a list of certificate stores that contain untrusted, intermediary certificate files awaiting validation, see “Configuring trusted ID evaluators on the server level” on page 3435.

6. Select a trusted ID evaluator from the Trusted ID evaluation reference field. This field specifies a reference to the Trusted ID evaluator class name that is defined in Trusted ID evaluators panel. The trusted ID evaluator is used for evaluating whether the received ID is trusted. If you select **None**, the trusted ID evaluator is not referenced in this token consumer configuration. To configure a trusted ID evaluator, see “Configuring trusted ID evaluators on the server level” on page 3435.
7. Select the **Verify nonce** option if a nonce is included in a user name token on the generator side. Nonce is a unique cryptographic number that is embedded in a message to help stop repeat, unauthorized attacks of user name tokens. The **Verify nonce** option is available if you specify a user name token for the token consumer and nonce is added to the user name token on the generator side.

8. Select the **Verify timestamp** option if a time stamp is included in the user name token on the generator side. The **Verify Timestamp** option is available if you specify a user name token for the token consumer and a time stamp is added to the user name token on the generator side.
9. Specify the local name of the value type for the integrated token. This entry specifies the local name of the value type for a security token that is referenced by the key identifier. This attribute is valid when **Key identifier** is selected as the key information type. To specify the key information type, see "Configuring the key information for the consumer binding using JAX-RPC on the server level" on page 3377. WebSphere Application Server has predefined value type local names for the user name token and the X.509 certificate security token. Enter one of the following local names for the user name token and the X.509 certificate security token. When you specify the following local names, you do not need to specify the URI of the value type:

   Username token
   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

   X.509 certificate token
   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

   X.509 certificates in a PKIPath
   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1

   A list of X.509 certificates and CRLs in a PKCS#7
   http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7

   Note: To specify Lightweight Third Party Authentication (LTPA) or token propagation (LTPA_PROPAGATION), you must specify both the value type local name and the Uniform Resource Identifier (URI). For LTPA, specify `LTPA` for the local name and `http://www.ibm.com/websphere/appserver/tokentype/5.0.2` for the URI. For LTPA token propagation, specify `LTPA_PROPAGATION` for the local name and `http://www.ibm.com/websphere/appserver/tokentype` for the URI.

   For example, when an X.509 certificate token is specified, you can use `http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3` for the local name. When you specify the local name of another token, you must specify a value type Qname. For example: uri=`http://www.ibm.com/custom`, localName=`CustomToken`
10. Specify the value type uniform resource identifier (URI) in the URI field. This entry specifies the namespace URI of the value type for a security token that is referenced by the key identifier. This attribute is valid when **Key identifier** is selected as the key information type on the Key information panel for the default generator. When you specify the token consumer for the user name token or an X.509 certificate security token, you do not need to specify this option. If you specify another token, you need to specify the URI of the QName for the value type.
11. Click **OK** and then **Save** to save the configuration. After saving the token generator configuration, you can specify a JAAS configuration for your token consumer.
12. Click the name of your token generator configuration.
13. Under Additional properties, click **JAAS configuration**.
14. Select a JAAS configuration from the JAAS configuration name field.

    The field specifies the name of the JAAS system for application login configuration. You can specify additional JAAS system and application configurations by clicking **Security** > **Global security**. Expand Java Authentication and Authorization Service, then click **Application logins** > **New** or **System logins** > **New**. Do not remove the predefined system or application login configurations. However, within these configurations, you can add module class names and specify the order in which WebSphere Application Server loads each module. WebSphere Application Server provides the following predefined JAAS configurations:

    ClientContainer
    This selection specifies the login configuration that is used by the client container

applications. The configuration uses the CallbackHandler application programming interface (API) that is defined in the deployment descriptor for the client container. To modify this configuration, see the JAAS configuration panel for application logins.

WSLogin
This selection specifies whether all of the applications can use the WSLogin configuration to perform authentication for the security run time. To modify this configuration, see the JAAS configuration panel for application logins.

DefaultPrincipalMapping
This selection specifies the login configuration that is used by Java 2 Connectors (J2C) to map users to principals that are defined in the J2C authentication data entries. To modify this configuration, see the JAAS configuration panel for application logins.

system.wssecurity.IDAssertion
This selection enables a Version 5.*x* application to use identity assertion to map a user name to a WebSphere Application Server credential principal. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.Signature
This selection enables a Version 5.*x* application to map a distinguished name (DN) in a signed certificate to a WebSphere Application Server credential principal. To modify this configuration, see the JAAS configuration panel for system logins.

system.LTPA_WEB
This selection processes login requests that are used by the web container such as servlets and JavaServer Pages (JSP) files. To modify this configuration, see the JAAS configuration panel for system logins.

system.WEB_INBOUND
This selection handles login requests for web applications, which include servlets and JavaServer Pages (JSP) files. This login configuration is used by WebSphere Application Server Version 5.1.1. To modify this configuration, see the JAAS configuration panel for system logins.

system.RMI_INBOUND
This selection handles logins for inbound Remote Method Invocation (RMI) requests. This login configuration is used by WebSphere Application Server Version 5.1.1. To modify this configuration, see the JAAS configuration panel for system logins.

system.DEFAULT
This selection handles the logins for inbound requests that are made by internal authentications and most of the other protocols except web applications and RMI requests. This login configuration is used by WebSphere Application Server Version 5.1.1. To modify this configuration, see the JAAS configuration panel for system logins.

system.RMI_OUTBOUND
This selection processes RMI requests that are sent outbound to another server when the `com.ibm.CSIOutboundPropagationEnabled` property is `true`. This property is set in the CSIv2 authentication panel. To access the panel, click **Security** > **Global security**. Under Authentication, expand **RMI/IIOP security** and click **CSIv2 outbound authentication**. To set the `com.ibm.CSIOutboundPropagationEnabled` property, select **Security attribute propagation**. To modify this JAAS login configuration, see the JAAS - System logins panel.

system.wssecurity.X509BST
This section verifies an X.509 binary security token (BST) by checking the validity of the certificate and the certificate path. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.PKCS7
This selection verifies an X.509 certificate with a certificate revocation list in a PKCS7 object. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.PkiPath
This section verifies an X.509 certificate with a public key infrastructure (PKI) path. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.UsernameToken
This selection verifies the basic authentication (user name and password) data. To modify this configuration, see the JAAS configuration panel for system logins.

system.wssecurity.IDAssertionUsernameToken
This selection enables Versions 6 and later applications to use identity assertion to map a user name to a WebSphere Application Server credential principal. To modify this configuration, see the JAAS configuration panel for system logins.

system.WSS_INBOUND
This selection specifies the login configuration for inbound or consumer requests for security token propagation using Web Services Security. To modify this configuration, see the JAAS configuration panel for system logins.

system.WSS_OUTBOUND
This selection specifies the login configuration for outbound or generator requests for security token propagation using Web Services Security. To modify this configuration, see the JAAS configuration panel for system logins.

None With this selection, you do not specify a JAAS login configuration.

15. Click **OK** and then **Save** to save the configuration.

Results

You have configured the token consumer at the server level.

What to do next

You must specify a similar token generator configuration for the server level.

Token consumer collection:

Use this page to view the token consumer. The information is used on the consumer side only to process the security token.

To view this administrative console page for the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Default Generator Bindings, click **Token consumers**.

To view this administrative console page for Version 6.*x* and later applications on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name*** .

3. Under Web Services Security Properties, you can access the signing information for the following bindings:
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. Under Required properties, click **Token consumers**.
   - For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Token consumers**.

Token consumer name:

Specifies the name of the token consumer configuration.

For example, the default X509 token consumer names can be either `con_enctcon` for encrypting or `con_signtcon` for signing. Or a custom token consumer name might be `sig_tcon` for signing.

Token consumer class name:

Specifies the name of the token consumer implementation class.

This class must implement the `com.ibm.wsspi.wssecurity.token.TokenConsumerComponent` interface.

Token consumer class name:

Specifies the name of the token consumer implementation class.

The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to validate (authenticate) the security token on the consumer side.

Token consumer configuration settings:

Use this page to specify the information for the token consumer. The information is used at the consumer side only to process the security token.

To view this administrative console page for the server level, complete the following steps:
1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** .
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under JAX-RPC Default Consumer Bindings, click **Token consumers** > ***token_consumer_name*** or click **New** to create a new token consumer.

To view this administrative console page for Version 6 and later applications on the application level, complete the following steps:
1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access the signing information for the following bindings:
   - For the Response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**. Under Required properties, click **Token consumers**.

- For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**. Under Required properties, click **Token consumers**.

4. Click **New** to specify a new configuration or click the name of an existing configuration to modify its settings.

Before specifying additional properties, specify a value in the Token consumer name, the Token consumer class name, and the Value type local name fields.

Token consumer name:

Specifies the name of the token consumer configuration.

For example, the default X509 token consumer names are either `con_enctcon` for encrypting or `con_signtcon` for signing. Or a custom, the token consumer name might be `sig_tcon` for signing.

Token consumer class name:

Specifies the name of the token consumer implementation class.

This class must implement the `com.ibm.wsspi.wssecurity.token.TokenConsumerComponent` interface.

Token consumer class name:

Specifies the name of the token consumer implementation class.

The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to validate (authenticate) the security token on the consumer side.

Part reference:

Specifies a reference to the name of the security token that is defined in the deployment descriptor.

On the application level, when the security token is not specified in the deployment descriptor, the Part reference field is not displayed.

Certificate path:

Specifies the trust anchor and the certificate store.

You can select the following options:

None If you select this option, the certificate path is not specified.

Trust any
If you select this option, any certificate is trusted. When the received token is incorporated, the certificate path validation is not processed.

Dedicated signing information
If you select this option, you can specify the trust anchor and the certificate store. When you select the trust anchor or the certificate store of a trusted certificate, you must configure the collection certificate store before setting the certificate path.

Trust anchor

You can specify a trust anchor for the following bindings on the following levels:

Table 317. Trust anchor binding settings. The trust anchor is used for signing messages.

Binding name	Server level or application level	Path
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Trust anchors**.

Certificate store

You can specify a certificate path configuration for the following bindings on the following levels:

Table 318. Certificate store binding settings. The certificate is used for signing messages.

Binding name	Server level or application level	Path
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Collection certificate store**.

Trusted ID evaluator reference:

Specifies the reference to the Trusted ID evaluator class name that is defined in the Trusted ID evaluators panel. The trusted ID evaluator is used for determining whether the received ID is trusted.

You can select the following options:

None If you select this option, the trusted ID evaluator is not specified.

Existing evaluator definition
If you select this option, you can select one of the configured trusted ID evaluators.

You can specify a certificate path configuration for the following bindings on the following levels:

Table 319. Trusted ID evaluator bindings settings. The trusted ID evaluator is used to determine if a received ID is trusted.

Binding name	Server level or application level	Path
Default consumer binding	Server level	1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***. 2. Under Security, click **JAX-WS and JAX-RPC security runtime**. **Note:** In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**. 3. Under Additional properties, click **Trusted ID evaluators**.

Binding evaluator definition
If you select this option, you can specify a new trusted ID evaluator and its class name.

When you select a trusted ID evaluator reference, you must configure the trusted ID evaluators before setting the token consumer.

The Trusted ID evaluator field is displayed in the default binding configuration and the application server binding configuration.

Verify nonce:

Specifies whether the nonce of the user name token is verified.

This option is displayed on the cell, server, and application levels. This option is valid only when the type of incorporated token is the user name token.

Verify timestamp:

Specifies whether the time stamp of user name token is verified.

This option is displayed on the cell, server, and application levels. This option is valid only when the type of incorporated token is the user name token.

Value type local name:

Specifies the local name of value type for the consumed token.

This product has predefined value type local names for the user name token and the X.509 certificate security token. Use the following local names for the user name token and the X.509 certificate security token. When you specify the following local names, you do not need to specify the Uniform Resource Identifier (URI) of the value type:

Username token
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

X509 certificate token
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3

 # X509 certificates in a PKIPath
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509PKIPathv1

A list of X509 certificates and CRLs in a PKCS#7
http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#PKCS7

Lightweight Third Party Authentication (LTPA)

LTPA_PROPAGATION

Important: For Lightweight Third Party Authentication (LTPA), the value type local name is `LTPA`. If you enter `LTPA` for the local name, you must specify the `http://www.ibm.com/websphere/appserver/tokentype/5.0.2` URI value in the Value type URI field as well. For LTPA token propagation, the value type local name is `LTPA_PROPAGATION`. If you enter `LTPA_PROPAGATION` for the local name, you must specify the `http://www.ibm.com/websphere/appserver/tokentype` URI value in the Value type URI field as well. For the other predefined value types (Username token, X509 certificate token, X509 certificates in a PKIPath, and a list of X509 certificates and CRLs in a PKCS#7), the value for the local name field begins with `http://`. For example, if you are specifying the username token for the value type, enter `http://docs.oasis-open.org/`

wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken in the value type local name field and then you do not need to enter a value in the value type URI field.

When you specify a custom value type for custom tokens, you can specify the local name and the URI of the Quality name (QName) of the value type. For example, you might specify Custom for the local name and http://www.ibm.com/custom for the URI.

Value type URI:

Specifies the namespace URI of the value type for the integrated token.

When you specify the token consumer for the user name token or the X.509 certificate security token, you do not need to specify this option. If you want to specify another token, specify the URI of the QName for the value type.

The application server provides the following predefined value type URIs:

- For the LTPA token: http://www.ibm.com/websphere/appserver/tokentype/5.0.2
- For the LTPA token propagation: http://www.ibm.com/websphere/appserver/tokentype

Configuring Web Services Security using JAX-RPC at the platform level

In the platform configuration, general properties and additional properties can be specified, and the default binding is included. You can configure security for web services at a platform level with a variety of tasks including configuring key locators, trust anchors, and the collection certificate at the generator, consumer binding, and sever levels.

Before you begin

best-practices: IBM WebSphere Application Server supports the Java API for XML-Based Web Services (JAX-WS) programming model and the Java API for XML-based RPC (JAX-RPC) programming model. JAX-WS is the next generation web services programming model extending the foundation provided by the JAX-RPC programming model. Using the strategic JAX-WS programming model, development of web services and clients is simplified through support of a standards-based annotations model. Although the JAX-RPC programming model and applications are still supported, take advantage of the easy-to-implement JAX-WS programming model to develop new web services applications and clients.

Besides the application-level constraints, there is a server-level Web Services Security (WSS) configuration called a *platform-level configuration*:

- These configurations are global for all applications and include some configurations only for WebSphere Application Server Version 5.*x* applications and some only for version 6.0.*x* applications.
- You can use the default binding as an application-level binding configuration so that applications do not have to define the binding in the application. There is only one set of default bindings that can be shared by multiple applications. This set is only available for WebSphere Application Server Version 6.*x* applications.

Therefore, binding configuration files can be specified at these levels: application and server. Each binding configuration overrides the next higher one. For any deployed application, the nearest configuration binding is applied. The visibility scope of the binding depends on where the file is located. If the binding is defined in an application, its visibility is scoped to that particular application. If it is located at the server level, the visibility scope is all applications that are deployed on that server.

About this task

To ensure Web Services Security at the platform level, you can configure:

- A nonce on the server level
- The key locator for the generator or consumer binding on the application level or at the server level
- Trust anchors for the generator or consumer binding on the application level or at the server level
- The collection certificate store for the generator or consumer binding on the application level or server level
- Trusted ID evaluators on the server level
- Hardware cryptographic devices for Web Services Security
- The `rrdSecurity.props` property file

Procedure

- To configure a nonce on the server level, see the steps in "Configuring a nonce on the server level"
- To configure the key locator for the generator binding on the application level, see the steps in "Configuring the key locator using JAX-RPC for the generator binding on the application level" on page 3405
- To configure the key locator for the consumer binding on the application level, see the steps in "Configuring the key locator using JAX-RPC for the consumer binding on the application level" on page 3412
- To configure the key locator on the server level, see the steps in "Configuring the key locator using JAX-RPC on the server level" on page 3414
- To configure trust anchors for the generator binding on the application level, see the steps in "Configuring trust anchors for the generator binding on the application level" on page 3416
- To configure trust anchors for the consumer binding on the application level, see the steps in "Configuring trust anchors for the consumer binding on the application level" on page 3420
- To configure trust anchors on the server level, see the steps in "Configuring trust anchors on the server level" on page 3422
- To configure the collection certificate store for the generator binding on the application level, see the steps in "Configuring the collection certificate store for the generator binding on the application level" on page 3423
- To configure the collection certificate store for the consumer binding on the application level, see the steps in "Configuring the collection certificate store for the consumer binding on the application level" on page 3431
- To configure the collection certificate on the server level, see the steps in "Configuring the collection certificate on the server level" on page 3433
- To configure trusted ID evaluators on the server level, see the steps in "Configuring trusted ID evaluators on the server level" on page 3435
- To enable hardware cryptographic devices for Web Services Security, see the steps in "Enabling hardware cryptographic devices for Web Services Security" on page 3440
- To work with the `rrdSecurity.props` file, see "rrdSecurity.props file" on page 3438

Results

By completing these steps, you have configured Web Services Security at the platform level.

Configuring a nonce on the server level:

You can configure nonce for the server by using the WebSphere Application Server administrative console.

About this task

Nonce is a randomly generated, cryptographic token that is used to prevent replay attacks of user name tokens that are used with SOAP messages. Typically, nonce is used with the user name token.

You can configure nonce at the application level and the server level. However, you must consider the order of precedence.

The following list shows the order of precedence:

1. Application level

   The application level settings for the nonce maximum age and nonce clock skew fields are specified through the additional properties.
2. Server level

If you configure nonce on the application level and the server level, the values that are specified for the application level take precedence over the values that are specified for the server level. Likewise, the values that are specified for the application level take precedence over the values specified for the server level. Complete the following steps to configure nonce on the server level:

Complete the following steps to configure a nonce on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers > Server Types > WebSphere application servers > *server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Specify a value, in seconds, for the **Nonce cache timeout** field. The value that is specified for the **Nonce cache timeout** field indicates how long the nonce remains cached before it is discarded. You must specify a minimum of 300 seconds. However, if you do not specify a value, the default is 600 seconds. This field is optional on the server level.
3. Specify a value, in seconds, for the **Nonce maximum age** field. The value that is specified for the **Nonce maximum age** field indicates how long the nonce is valid. You must specify a minimum of 300 seconds, but the value cannot exceed the number of seconds that is specified for the **Nonce cache timeout** field. If you do not specify a value, the default is 300 seconds. This field is optional on the server level.
4. Specify a value, in seconds, for the **Nonce clock skew** field. The value that is specified for the **Nonce clock skew** field specifies the amount of time, in seconds, to consider when the message receiver checks the freshness of the value. Consider the following information when you set this value:
   - Difference in time between the message sender and the message receiver, if the clocks are not synchronized.
   - Time that is needed to encrypt and transmit the message.
   - Time that is needed to get through network congestion.

   At a minimum, you must specify 0 seconds in this field. However, the maximum value cannot exceed the number of seconds indicated in the Nonce maximum age field. If you do not specify a value, the default is 0 seconds. This field is optional on the server level.
5. Optional: For WebSphere Application Server, Network Deployment only, select **Distribute nonce caching**. This option enables you to distribute the caching for a nonce using a Data Replication Service (DRS). In previous releases of WebSphere Application Server, the nonce was cached locally. By selecting this option, the nonce is propagated to other servers in your environment. However, the nonce might be subject to a one-second delay in propagation and subject to any network congestion.

6. Restart the server. If you change the nonce cache timeout value and do not restart the server, the change is not recognized by the server.

Distributing nonce caching to servers in a cluster:

Distributed nonce caching enables you to distribute the cache for a nonce to different servers in a cluster.

Before you begin

Before configuring distributed nonce caching, configure cache replication.

About this task

In previous releases of WebSphere Application Server, the nonce was cached locally. To use this feature, you must complete the following actions:

Procedure

1. Verify that you created an appropriate domain setting when you form a cluster.
2. Verify that replication domain is properly secured. The nonce cache is crucial to the integrity of the nonce validation process. If the nonce cache is compromised, then you cannot trust the result of the validation process.
3. In the administrative console for the server level, select the **Distribute nonce caching** option. You can enable the option by completing the following steps:
   a. Click **Security > Web services**.
   b. Select the **Distribute nonce caching** option.
4. Restart the servers within your cluster.

Results

When you select the **Distribute nonce caching** option in the administrative console, the nonce is propagated to other servers in your environment. However, the nonce might be subject to a one-second delay in propagation and subject to any network congestion.

What to do next

For more information on distributed nonce caching, see Web Services Security enhancements.

Configuring the key locator using JAX-RPC for the generator binding on the application level:

The key locator information for the default generator specifies which key locator implementation is used to locate the key to be used for signature and encryption information. The key locator information for the generator specifies which key locator implementation is used to locate the key to be used for signature validation or encryption.

About this task

WebSphere Application Server provides default values for the bindings. However, you must modify the defaults for a production environment.

Complete the following steps to configure the key locator for the generator binding on the application level:

Procedure

1. Locate the encryption information configuration panel in the administrative console.

a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
b. Under Manage modules, click ***URI_name***.
c. Under Web Services Security Properties you can access the key information for the request generator and response generator bindings.
   - For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
d. Under Additional properties, click **Key locators**.
e. Click **New** to create a key locator configuration, select the box next to the configuration and click **Delete** to delete an existing configuration, or click the name of an existing key locator configuration to edit its settings. If you are creating a new configuration, enter a unique name in the **Key locator name** field. For example, you might specify `gen_keyloc`.

2. Specify a class name for the key locator class implementation in the **Key locator class name** field. The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to create the security token on the generator side. Specify a class name according to the requirements of the application. For example, if the application requires that the key is read from a keystore file, specify the com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator implementation. WebSphere Application Server supports the following default key locator class implementations for Versions 6.0.*x* and later applications that are available to use with the request generator or response generator:

   com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator
   This implementation locates and obtains the key from the specified keystore file.

   com.ibm.wsspi.wssecurity.keyinfo.SignerCertKeyLocator
   This implementation uses the public key from the signer certificate and is used by the response generator.

3. Specify the keystore password, the keystore location, and the keystore type. Keystore files contain public and private keys, root certificate authority (CA) certificates, the intermediate CA certificate, and so on. Keys retrieved from the keystore are used to sign and validate or encrypt and decrypt messages or message parts. If you specified the com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator implementation for the key locator class implementation, you must specify a keystore password, location, and type.
   a. Specify a password in the keystore **Password** field. This password is used to access the keystore file.
   b. Specify the location of the keystore file in the keystore **Path** field.
   c. Select a keystore type from the **Type** field. The Java Cryptography Extension (JCE) that is used by IBM supports the following keystore types:

      JKS Use this option if you are not using Java Cryptography Extensions (JCE) and if your keystore file uses the Java Keystore (JKS) format.

      JCEKS
      Use this option if you are using Java Cryptography Extensions.

      JCERACFKS
      Use JCERACFKS if the certificates are stored in a SAF key ring (z/OS only).

      PKCS11KS (PKCS11)
      Use this format if your keystore uses the PKCS#11 file format. Keystores using this format might contain RSA keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

      PKCS12KS (PKCS12)
      Use this option if your keystore uses the PKCS#12 file format.

WebSphere Application Server provides some sample keystore files in the ${USER_INSTALL_ROOT}/etc/ws-security/samples directory. For example, you might use the enc-receiver.jceks keystore file for encryption keys. The password for this file is Storepass and the type is JCEKS.

Restriction: Do not use the sample keystore files in a production environment. These samples are provided for testing purposes only.

4. Click **OK** and then click **Save** to save the configuration.
5. Under Additional properties, click **Keys**.
6. Click **New** to create a key configuration, select the box next to the configuration and click **Delete** to delete an existing configuration, or click the name of an existing key configuration to edit its settings. This entry specifies the name of the key object within the keystore file. If you are creating a new configuration, enter a unique name in the **Key name** field. For digital signatures, the key name is used by the request generator or the response generator signing information to determine which key is used to digitally sign the message.

   You must use a fully qualified distinguished name for the key name. For example, you might use CN=Bob,O=IBM,C=US.

   Important: Do not use the sample key files in a production environment. These samples are provided for testing purposes only.
7. Specify an alias in the **Key alias** field. The key alias is used by the key locator to search for key objects in the keystore.
8. Specify a password in the **Key password** field. The password is used to access the key object within the keystore file.
9. Click **OK** and **Save** to save the configuration.

Results

You have configured the key locator for the generator binding at the application level.

What to do next

You must specify a similar key information configuration for the consumer.

Key locator collection:

Use this page to view a list of key locator configurations that retrieve keys from the keystore for digital signature and encryption. A key locator must implement the com.ibm.wsspi.wssecurity.config.KeyLocator interface.

To view this administrative console page for the key locator collection on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Key locators**.

To use this administrative console page for the key locator collection on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.

3. Under Web Services Security Properties, you can access key locators for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Key locators**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Key locators**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Key locators**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Key locators**.
4. Under Additional properties, you can access key locators for the following bindings:
   - For the Request sender, click **Web services: Client security bindings**. Under Request sender binding, click **Edit** > **Key locators**.
   - For the Request receiver, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit** > **Key locators**.
   - For the Response sender, click **Web services: Server security bindings**. Under Response sender binding, click **Edit** > **Key locators**.
   - For the Response receiver, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit** > **Key locators**.

Tip: The bindings for a Version 6.x. or later application has a link that says **Edit custom**.

Using this **Key locator collection** panel, complete the following steps:

1. Specify a key locator name and a key locator class name on the panel.
2. Save your changes by clicking **Save** in the messages section at the top of the administrative console. The administrative console home panel is displayed.
3. After saving your changes, update the Web Services Security run time with the default binding information by clicking **Update runtime**. When you click **Update runtime**, the configuration changes made to the other Web services also are updated in the Web Services Security run time.
4. After you define key locators, click the key locator name to specify additional properties and keys under **Additional Properties**.

Key locator name:

Specifies the unique name of the key locator.

Key locator class name:

Specifies the class name of the key locator, which retrieves the key that is used for digital signing and encryption.

Key locator configuration settings:

Use this page to specify the settings for a key locator configuration. The key locators retrieve keys from the keystore file for digital signature and encryption. This product enables you to plug in a custom key locator configuration.

To view this administrative console page for the key locator collection on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

3. Under Additional properties, click **Key locators**.
4. Click **New** to create a new configuration or click the name of a configuration to modify its settings.

To use this administrative console page for the key locator collection on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.
3. Under Web Services Security properties, you can access key locators for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Key locators**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Key locators**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Key locators**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Key locators**.
4. Click **New** to create a new configuration or click the name of a configuration to modify its settings.

Key locator name:

Specifies the name of the key locator.

Data type String

Key locator class name:

Specifies the name for the key locator class implementation.

Key locators that are associated with Versions 6 and later applications must implement the `com.ibm.wsspi.wssecurity.keyinfo.KeyLocator` interface. This product provides the following default key locator class implementations for Versions 6 and later applications:

com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator
This implementation locates and obtains the key from the specified keystore file.

com.ibm.wsspi.wssecurity.keyinfo.SignerCertKeyLocator
This implementation uses the public key from the certificate of the signer. This class implementation is used by the response generator.

This property is for the JAX-RPC programming model only. To implement signer certificate encryption for the JAX-WS programming model, set a custom property on the callback handler for the encryption token generator. For more information, read the topic Callback handler settings.

com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator
This implementation uses the X.509 security token from the sender message for digital signature validation and encryption. This class implementation is used by the request consumer and the response consumer.

Data type String

Keystore: Specifies information about the key store that is used by this key locator configuration.

None Use this option if a key store is not required to be specified for this key locator configuration.

Predefined keystore
Use this option if you want to specify a predefined keystore for this key locator configuration.

User-defined keystore
Use this option if you want to specify a user-defined key store for this key locator configuration.

Keystore configuration name:

Specifies the name of the key store configuration that is defined in the keystore settings in secure communications.

The keystore configuration name is located under the **Predefined keystore** field, which is located under the **Keystore** section of the page.

Data type String

Keystore password:

Specifies the password that is used to access the keystore file.

The keystore password is located under the **User-defined keystore** field, which is located under the **Keystore** section of the page.

Data type String

Keystore path:

Specifies the location of the keystore file.

The path is located under the **User-defined keystore** field, which is located under the **Keystore** section of the page.

Data type String

Keystore type:

Specifies the type of keystore file.

The type is located under the **User-defined keystore** field, which is located under the **Keystore** section of the page.

JKS Use this option if you are not using Java Cryptography Extensions (JCE) and if your keystore file uses the Java Keystore (JKS) format.

JCEKS
Use this option if you are using Java Cryptography Extensions.

PKCS11KS (PKCS11)
Use this format if your keystore file uses the PKCS#11 file format. Keystores files that use this format might contain Rivest Shamir Adleman (RSA) keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

PKCS12KS (PKCS12)
Use this option if your keystore file uses the PKCS#12 file format.

Default JKS
Range JKS, JCEKS, PKCS11KS (PKCS11), PKCS12KS (PKCS12)

Web Services Security property collection:

Use this page to a view a list of additional properties for the configuration.

You can view a Web Services Security property collection panel at the cell level. Complete the following steps to view one of these administrative console pages:

1. Click **Security** > **JAX-WS and JAX-RPC security runtime**.
2. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Properties**.
3. Click **New** to create a new property.
4. Click **Delete** to a delete a property that you specified previously.

Property name:

Specifies the name of the property.

Property value:

Specifies the value for the property.

Web Services Security property configuration settings:

Use this page to configure additional security properties.

You can view a Web Services Security property configuration settings panel at the cell level. Complete the following steps to view one of these administrative console pages:

1. Click **Security** > **JAX-WS and JAX-RPC security runtime**.
2. Under JAX-RPC Default Generator Bindings or JAX-RPC Default Consumer Bindings, click **Properties** > **New**.

Property Name:

Specifies the name of the property.

Data type: String

Property Value:

Specifies the value for the property.

Data type: String

The following table lists the properties that you can configure by using the Web Services Security property panels.

Table 320. Property configuration settings. The properties are used to secure web services.

Configuration panel name	Property name	Property value	Description
JAAS configuration	com.ibm.wsspi.wssecurity.token.X509.issuerName	Specify the SubjectDN or the IssuerDN of the issuer for the X.509 certificate.	This property is used to specify the issuer of the certificate in the token consumer component.
JAAS configuration	com.ibm.wsspi.wssecurity.token.X509.issuerSerial	Specify the serial number of the X.509 certificate.	This property is used to specify the serial number of the certificate in the token consumer component.
Key information	com.ibm.wsspi.wssecurity.keyinfo.EncodingNS	Specify the namespace Uniform Resource Identifier (URI) for the qualified name (QName).	This property is used to specify the namespace URI part of the QName that represents the encoding method.
Properties	com.ibm.ws.wssecurity.handler.hardwareCacheEntryRefreshHours	Specify a numeric value from 1 to 24 that represents the number of hours that a temporary key is valid.	This property is used to specify the amount of time before a key is retranslated. Temporary keys outside the keystore typically expire in a short period of time, measured in days or hours. If the server is configured to use a hardware acceleration card, but not the hardware keystore, you can configure it to translate the temporary keys periodically before they expire. If this property is not set, a key will be retranslated after 8 hours. Setting this value to 0 disables retranslation.
Request generator and Response generator	com.ibm.wsspi.wssecurity.timestamp.SOAPHeaderElement	Specify `1` or `true`.	This property is used with the **Add nonce** option to set the mustUnderstand flag in the deployment descriptor.
Request generator and Response generator	com.ibm.wsspi.wssecurity.timestamp.dialect		
Signing information	com.ibm.wsspi.wssecurity.dsig.dumpPath	Specify the path used to locate the output file.	This property is used to specify an output file for dumping the target UTF-8 binary data before signing and verifying messages.
Token generator	com.ibm.wsspi.wssecurity.token.username.timestampExpires	Specify `1` or `true`.	This property is used to specify an expiration date for the user name token.
Transform algorithms	com.ibm.wsspi.wssecurity.dsig.XPathExpression	`not(ancestor-or-self::*[namespace-uri()='http://www.w3.org/2000/09/xmldsig#' and local-name()='Signature'])`	This property is used with this algorithm: `http://www.w3.org/TR/1999/REC-xpath-19991116`

Configuring the key locator using JAX-RPC for the consumer binding on the application level:

The key locator information for the consumer at the application level specifies which key locator implementation is used. The key locator implementation locates the key to be used to validate the digital signature or the encryption information by the application.

About this task

Complete the following steps to configure the key locator for the consumer binding on the application level:

Procedure

1. Locate the key locator configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties, you can access the key information for the request consumer and response consumer bindings.

- For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
- For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.

d. Under Additional properties, click **Key locators**.

e. Click **New** to create a key locator configuration, click **Delete** and select the box next to the configuration to delete an existing configuration, or click the name of an existing key locator configuration to edit its settings. If you are creating a new configuration, enter a unique name in the **Key locator name** field. For example, you might specify `klocator`.

2. Specify a name for the key locator class implementation. The Java Authentication and Authorization Service (JAAS) Login Module implementation is used to validate (authenticate) the security token on the consumer side. Specify a class name according to the requirements of the application. For example, if the application requires that the key is read from a keystore file, specify the com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator implementation. WebSphere Application Server provides the following default key locator class implementations for Version 6.0.*x* applications that are available to use with the request consumer or response consumer:

   com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator
   This implementation locates and obtains the key from the specified keystore file.

   com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator
   This implementation uses the X.509 security token from the sender message for digital signature validation and encryption. This class implementation is used by the request consumer and the response consumer.

3. Specify the keystore password, the keystore location, and the keystore type. Keystore files contain public and private keys, root certificate authority (CA) certificates, the intermediate CA certificate, and so on. Keys that are retrieved from the keystore files are used to sign and validate or encrypt and decrypt messages or message parts. If you specified the com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator implementation for the key locator class implementation, you must specify a keystore password, location, and type.

   a. Specify a password in the keystore **Password** field. This password is used to access the keystore file.

   b. Specify the location of the keystore file in the keystore **Path** field.

   c. Select a keystore type from the keystore **Type** field. The Java Cryptography Extension (JCE) that is used by IBM supports the following keystore types:

      JKS Use this option if you are not using Java Cryptography Extensions (JCE) and if your keystore file uses the Java Keystore (JKS) format.

      JCEKS
      Use this option if you are using Java Cryptography Extensions.

      PKCS11KS (PKCS11)
      Use this format if your keystore file uses the PKCS#11 file format. Keystore files that use this format might contain RSA keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

      PKCS12KS (PKCS12)
      Use this option if your keystore uses the PKCS#12 file format.

      WebSphere Application Server provides some sample keystore files in the `${USER_INSTALL_ROOT}/etc/ws-security/samples` directory. For example, you might use the `enc-receiver.jceks` keystore file for encryption keys. The password for this file is `Storepass` and the type is `JCEKS`.

      Attention: Do not use these keystore files in a production environment. These samples are provided for testing purposes only.

4. Click **OK** and **Save** to save the configuration.
5. Under Additional properties, click **Keys**.

6. Click **New** to create a key configuration, click **Delete** and select the box next to the configuration to delete an existing configuration, or click the name of an existing key configuration to edit its settings. This entry specifies the name of the key object within the keystore file. If you are creating a new configuration, enter a unique name in the **Key name** field.

   It is recommended that you use a fully qualified distinguished name for the key name. For example, you might use `CN=Bob,O=IBM,C=US`.
7. Specify an alias in the **Key alias** field. The key alias is used by the key locator to search for key objects in the keystore file.
8. Specify a password in the **Key password** field. The password is used to access the key object within the keystore file.
9. Click **OK** and then click **Save** to save the configuration.

Results

You have configured the key locator for the consumer binding at the application level.

What to do next

You must specify a similar key information configuration for the generator.

Configuring the key locator using JAX-RPC on the server level:

The key locator information for the default generator bindings specifies which key locator implementation is used to locate the key for signature and encryption information if these bindings are not defined at the application level.

About this task

The key locator information for the default consumer bindings specifies which key locator implementation is used to locate the key that is used for signature validation or decryption if these bindings are not defined at the application level. WebSphere Application Server provides default values for the bindings. However, you must modify the defaults for a production environment.

Complete the following steps to configure the key locator on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Additional properties, click **Key locator**. You can configure the key locator configurations for both the default generator and the default consumer in this location.
3. Click one of the following to work with the key locator configurations:

   New To create a key locator configuration. Enter a unique name for the key locator configuration in the **Key locator name** field. For example, you might specify `sig_klocator`.

   Delete To delete an existing configuration

   an existing key locator configuration
   To edit the settings of an existing configuration.

4. Specify a name for the key locator class implementation in the **Key locator class name** field. The key locators that are associated with Version 6.0.*x* applications must implement the `com.ibm.wsspi.wssecurity.keyinfo.KeyLocator` interface.

   Note: This interface is valid only for JAX-RPC applications. For JAX-WS applications, the Java Authentication and Authorization Service (JAAS) Login Module implementation is used to create the security token on the generator side and to validate (authenticate) the security token on the consumer side.

   WebSphere Application Server provides the following default key locator class implementations for Version 6.0.*x* applications:

   com.ibm.wsspi.wssecurity.keyinfo.KeyStoreLeyLocator
   This implementation locates and obtains the key from a specified keystore file.

   com.ibm.wsspi.wssecurity.keyinfo.SignerCertKeyLocator
   This implementation uses the public key from the certificate of the signer. This class implementation is used by the response generator.

   com.ibm.wsspi.wssecurity.keyinfo.X509TokenKeyLocator
   This implementation uses the X.509 security token from the sender message for digital signature validation and encryption. This class implementation is used by the request consumer and the response consumer.

   For example, you might specify the `com.ibm.wsspi.wssecurity.keyinfo.KeyStoreLeyLocator` implementation if you need the configuration to be the key locator for signing information.
5. Specify the keystore password, the keystore location, and the keystore type. Keystore files contain public and private keys, root certificate authority (CA) certificates, the intermediate CA certificate, and so on. Keys that are retrieved from the keystore file are used to sign and validate or encrypt and decrypt messages or message parts. If you specified the `com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator` implementation for the key locator class implementation, you must specify a key store password, location, and type.
   a. Specify a password in the **Key store password** field. This password is used to access the keystore file.
   b. Specify the location of the keystore file in the **Key store path** field.
   c. Select a keystore type from the **Key store type** field. The Java Cryptography Extension (JCE) that is used supports the following key store types:

      JKS Use this option if you are not using Java Cryptography Extensions (JCE) and if your keystore file uses the Java Keystore (JKS) format.

      JCEKS
      Use this option if you are using Java Cryptography Extensions.

      PKCS11
      Use this format if your keystore file uses the PKCS#11 file format. Keystore files that use this format might contain Rivest Shamir Adleman (RSA) keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

      PKCS12
      Use this option if your keystore file uses the PKCS#12 file format.

      WebSphere Application Server provides some sample keystore files in the `${USER_INSTALL_ROOT}/etc/ws-security/samples` directory. For example, you might use the `enc-receiver.jceks` keystore file for encryption keys. The password for this file is `storepass` and the type is `JCEKS`.

      Restriction: Do not use these keystore files in a production environment. These samples are provided for testing purposes only.
6. Click **OK** and **Save** to save the configuration.
7. Under Additional properties, click **Keys**.

8. Click one of the following to work with the key configurations:

   New To create a key configuration. Enter a unique name in the **Key name** field. You must use a fully qualified distinguished name for the key name. For example, you might use `CN=Bob,O=IBM,C=US`.

   Delete To delete an existing configuration.

   an existing key configuration
   To edit the settings of the existing configuration.

   This entry specifies the name of the key object within the keystore file.
9. Specify an alias in the **Key alias** field. The key alias is used by the key locator to search for key objects in the keystore file.
10. Specify a password in the **Key password** field. The password is used to access the key object within the keystore file.
11. Click **OK** and then click **Save** to save the configuration.

Results

You have configured the key locator for the server level.

What to do next

Configure the key information for the default generator and the default consumer bindings that reference this key locator.

Configuring trust anchors for the generator binding on the application level:

A *trust anchor* specifies key stores that contain trusted root certificates, which validate the signer certificate. These key stores are used by the request generator and the response generator (when web services are acting as client) to generate the signer certificate for the digital signature. You can configure trust anchors for the generator binding at the application level by using the administrative console.

Before you begin

You can configure a trust anchor using an assembly tool or the administrative console. This task describes how to configure the application-level trust anchor using the administrative console. For more information on assembly tools, see the related information.

About this task

The keystores are critical to the integrity of the digital signature validation. If they are tampered with, the result of the digital signature verification is doubtful and comprised. Therefore, it is recommended that you secure these keystores. The binding configuration that is specified for the request generator must match the binding configuration for the response generator.

The trust anchor configuration for the request generator on the client must match the configuration for the request consumer on the server. Also, the trust anchor configuration for the response generator on the server must match the configuration for the response consumer on the client.

Complete the following steps to configure trust anchors for the generator binding on the application level:

Procedure

1. Locate the trust anchor panel in the administrative console.

a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
b. Under Manage modules, click ***URI_name***.
c. Under Web Services Security Properties you can access the trust anchor configuration for the following bindings:
   - For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
   - For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
d. Under Additional properties, click **Trust anchors**.
e. Click **New** to create a trust anchor configuration, click **Delete** to delete an existing configuration, or click the name of an existing trust anchor configuration to edit its settings. If you are creating a new configuration, enter a unique name in the **Trust anchor name** field.

2. Specify the keystore password, the keystore location, and the keystore type. Key store files contain public and private keys, root certificate authority (CA) certificates, the intermediate CA certificate, and so on. Keys retrieved from the keystore are used to sign and validate or encrypt and decrypt messages or message parts. If you specified the `com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator` implementation for the key locator class implementation, you must specify a key store password, location, and type.
   a. Specify a password in the **Key store password** field. This password is used to access the keystore file.
   b. Specify the location of the key store file in the **Key store path** field.
   c. Select a keystore type from the **Key store type** field. The Java Cryptography Extension (JCE) used by IBM supports the following key store types:

      JKS Use this option if you are not using Java Cryptography Extensions (JCE) and if your keystore file uses the Java Keystore (JKS) format.

      JCEKS
      Use this option if you are using Java Cryptography Extensions.

      PKCS11KS (PKCS11)
      Use this format if your keystore uses the PKCS#11 file format. Keystores using this format might contain RSA keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

      PKCS12KS (PKCS12)
      Use this option if your keystore uses the PKCS#12 file format.

      WebSphere Application Server provides some sample keystore files in the following directory, using the *USER_INSTALL_ROOT* variable:

      For example, you might use the `enc-receiver.jceks` keystore file for encryption keys. The password for this file is `Storepass` and the type is `JCEKS`.

      Restriction: Do not use these keystore files in a production environment. These samples are provided for testing purposes only.

Results

This task configures trust anchors for the generator binding at the application level.

What to do next

You must specify a similar trust anchor configuration for the consumer.

Trust anchor collection:

Use this page to view a list of keystore objects that contain trusted root certificates. These objects are used for certificate path validation of incoming X.509-formatted security tokens. Keystore objects within trust anchors contain trusted root certificates that are used by the CertPath API to validate the trust of a certificate chain.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To create the keystore file, use the keytool utility. The keytool utility is available using the QShell Interpreter.

To view this administrative console page for trust anchors on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Trust anchors**.

To view this administrative console page for trust anchors on the application level,

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access trust anchors information for the following bindings:
   - For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
4. Under Additional properties, you can access the trust anchors information for the following bindings:
   - For the Response receiver binding, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**.
   - For the Request receiver binding, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**.
5. Under Additional properties, click **Trust anchors**.

If you click **Update runtime**, the Web Services Security run time is updated with the default binding information, which is contained in the `ws-security.xml` file that was previously saved. If you make changes on this panel, you must complete the following steps:

1. Save your changes by clicking **Save** at the top of the administrative console. When you click **Save**, you are returned to the administrative console home panel.
2. Return to the Trust anchors collection panel and click **Update runtime**. When you click **Update runtime**, the configuration changes made to the other web services also are updated in the Web Services Security run time.

Trust anchor name:

Specifies the unique name that is used to identify the trust anchor.

Key store path:

Specifies the location of the keystore file that contains the trust anchors.

Key store type:

Specifies the type of keystore file.

The value for this field is **JKS**, **JCEKS**, **JCERACFKS** (z/OS only), **JCE4758RACFKS** (z/OS only), **PKCS11KS (PKCS11)**, or **PKCS12KS (PKCS12)**.

Trust anchor configuration settings:

Use this information to configure a trust anchor. Trust anchors point to keystores that contain trusted root or self-signed certificates. This information enables you to specify a name for the trust anchor and the information that is needed to access a keystore. The application binding uses this name to reference a predefined trust anchor definition in the binding file (or the default).

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To view this administrative console page for trust anchors on the server level, complete the following steps:

1. Click **Servers > Server Types > WebSphere application servers > *server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Trust anchors**.
4. Click **New** to create a trust anchor or click the name of an existing configuration to modify its settings.

To view this administrative console page for trust anchors on the application level,

1. Click **Applications > Application Types > WebSphere enterprise applications > *application_name***.
2. Under Modules, click **Manage modules > *URI_name***.
3. Under Web Services Security Properties, you can access trust anchors information for the following bindings:
   - For the Response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   - For the Request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
4. Under Additional properties, you can access the trust anchors information for the following bindings:
   - For the Response receiver binding, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**.
   - For the Request receiver binding, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit**.
5. Under Additional properties, click **Trust anchors**.
6. Click **New** to create a trust anchor or click the name of an existing configuration to modify its settings.

Trust anchor name:

Specifies the unique name that is used by the application binding to reference a predefined trust anchor definition in the default binding.

Key store configuration name:

Specifies the name of the key store configuration defined in the keystore settings in secure communications.

Key store password:

Specifies the password that is needed to access the key store file.

Key store path:

Specifies the location of the keystore file.

Use ${*USER_INSTALL_ROOT*} as this path expands to the WebSphere Application Server path on your machine.

Key store type:

Specifies the type of keystore file.

Choose from the following options:

JKS Use this option if you are not using Java Cryptography Extensions (JCE).

JCEKS
Use this option if you are using Java Cryptography Extensions.

PKCS11KS (PKCS11)
Use this format if your keystore uses the PKCS#11 file format. Keystores that use this format might contain Rivest Shamir Adleman (RSA) keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

PKCS12KS (PKCS12)
Use this option if your keystore uses the PKCS#12 file format.

Default	JKS
Range	JKS, JCEKS, PKCS11KS (PKCS11), PKCS12KS (PKCS12)

Configuring trust anchors for the consumer binding on the application level:

You can configure trust anchors for the consumer binding at the application level.

About this task

This article does not describe how to configure trust anchors at the server or cell level. Trust anchors that are defined at the application level have a higher precedence over trust anchors that are defined at the server or cell level. For more information on creating and configuring trust anchors on the server or cell level, see “Configuring trust anchors on the server level” on page 3422.

You can configure a trust anchor at the application level using an assembly tool or the administrative console. This article describes how to configure the application-level trust anchor using the administrative console.

A trust anchor specifies key stores that contain trusted root Certificate Authority (CA) certificates, which validate the signer certificate. These keystores are used by the request consumer (as defined in the `ibm-webservices-bnd.xmi` file) and the response consumer (as defined in the `ibm-webservicesclient-bnd.xmi` file when a web service is acting as a client) to validate the X.509 certificate in the SOAP message. The keystores are critical to the integrity of the digital signature validation. If the keystores are tampered with, the result of the digital signature verification is doubtful and comprised. Therefore, it is recommended that you secure these keystores. The binding configuration specified for the request consumer in the `ibm-webservices-bnd.xmi` file must match the binding configuration for the response consumer in the `ibm-webservicesclient-bnd.xmi` file. The trust anchor configuration for the request

consumer on the server side must match the request generator configuration on the client side. Also, the trust anchor configuration for the response consumer on the client side must match the response generator configuration on the server side.

Complete the following steps to configure trust anchors for the consumer binding on the application level:

Procedure

1. Locate the trust anchor panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties you can access the trust anchor configuration for the following bindings:
      - For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom**.
      - For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   d. Under Additional properties, click **Trust anchors**.
   e. Click one of the following to work with trust anchor configuration:

      New To create a trust anchor configuration. Enter a unique name in the Trust anchor name field.

      Delete To delete the existing configuration selected in the box next to the configuration.

      an existing trust anchor configuration
      To edit the settings of an existing trust anchor configuration.
2. Specify the keystore password, the keystore location, and the keystore type. A trust anchor keystore file contains the trusted root Certificate Authority (CA) certificates that are used for validating the X.509 certificate that is used in digital signature or XML encryption.
   a. Specify a password in the Key store password field. This password is used to access the keystore file.
   b. Specify the location of the keystore file in the Key store path field.
   c. Select a keystore type from the Key store type field. The Java Cryptography Extension (JCE) that is used by IBM supports the following keystore types:

      JKS Use this option if you are not using Java Cryptography Extensions (JCE) and if your keystore file uses the Java Keystore (JKS) format.

      JCEKS
      Use this option if you are using Java Cryptography Extensions.

      PKCS11KS (PKCS11)
      Use this format if your keystore file uses the PKCS#11 file format. Keystore files that use this format might contain RSA keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

      PKCS12KS (PKCS12)
      Use this option if your keystore file uses the PKCS#12 file format.

      WebSphere Application Server provides some sample keystore files in the following directory, using the *USER_INSTALL_ROOT* variable:

      For example, you might use the `enc-receiver.jceks` keystore file for encryption keys. The password for this file is `storepass` and the type is `JCEKS`.

      Restriction: Do not use these keystore files in a production environment. These samples are provided for testing purposes only.

Results

You have configured trust anchors for the consumer binding at the application level.

What to do next

You must specify a similar trust anchor information for the generator.

Configuring trust anchors on the server level:

You can configure a list of keystore objects that contain trusted root certificates to be used for certificate path validation of incoming X.509-formatted security tokens.

Before you begin

Prior to completing the steps to configure trust anchors, you must create the keystore file using the keytool utility. The keytool utility is available using the QShell Interpreter.

About this task

This task provides the steps that are needed to configure a list of keystore objects that contain trusted root certificates. These objects are used for certificate path validation of incoming X.509-formatted security tokens. Keystore objects within trust anchors contain trusted root certificates that are used by the CertPath application programming interface (API) to determine whether to trust a certificate chain.

Complete the following steps to configure the trust anchors on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Additional properties, click **Trust anchors**.
3. Click one of the following to work with trust anchor configuration:

   New To create a trust anchor configuration. Enter a unique name for the trust anchor in the Trust anchor name field.

   Delete To delete an existing configuration.

   an existing trust anchor configuration
   To edit the settings for an existing trust anchor.
4. Specify a password in the Key store password field that is used to access the keystore file.
5. Specify the absolute location of the keystore file in the **Key store path** field. It is recommended that you use the *USER_INSTALL_ROOT* variable as a portion of the keystore path. To change this predefined variable, click **Environment** > **WebSphere variables**. The *USER_INSTALL_ROOT* variable might display on the second page of variables.
6. Specify the type of keystore file in the key store type field. WebSphere Application Server supports the following keystore types:

   JKS Use this option if you are not using Java Cryptography Extensions (JCE) and your keystore file uses the Java Key Store (JKS) format.

   JCEKS
   Use this option if you are using Java Cryptography Extensions.

PKCS11KS (PKCS11)
Use this option if your keystore file uses the PKCS#11 file format. Keystore files that use this format might contain Rivest Shamir Adleman (RSA) keys on cryptographic hardware or might encrypt keys that use cryptographic hardware to ensure protection.

PKCS12KS (PKCS12)
Use this option if your keystore file uses the PKCS#12 file format.

7. Click **OK** and **Save** to save your configuration.

Results

You have configured trust anchors at the server level.

Configuring the collection certificate store for the generator binding on the application level:

You can configure a collection certificate for the generator bindings on the application level.

About this task

A *collection certificate store* is a collection of non-root, certificate authority (CA) certificates and certificate revocation lists (CRLs). This collection of CA certificates and CRLs is used to check for a valid signature in a digitally signed SOAP message.

Complete the following steps to configure a collection certificate for the generator bindings on the application level:

Procedure

1. Locate the collection certificate store configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security Properties, you can access the key information for the request generator and response generator bindings.
      - For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
      - For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
   d. Under Additional properties, click **Collection certificate store**.
2. Specify the Certificate store name. Click **New** to create a collection certificate store configuration, select the box next to the configuration and click **Delete** to delete an existing configuration, .or click the name of an existing collection certificate store configuration to edit its settings. If you are creating a new configuration, enter a name in the Certificate store name field.

   The name of the collection certificate store must be unique to the level of the application server. For example, if you create the collection certificate store for the application level, the store name must be unique to the application level. The name that is specified in the Certificate store name field is used by other configurations to refer to a predefined collection certificate store. WebSphere Application Server searches for the collection certificate store based on proximity.

   For example, if an application binding refers to a collection certificate store named `cert1`, the Application Server searches for `cert1` at the application level before searching the server level.
3. Specify a certificate store provider in the Certificate store provider field. WebSphere Application Server supports the IBMCertPath certificate store provider. To use another certificate store provider, you must define the provider implementation in the provider list within the *`profile_root`*`/properties/java.security` file. However, make sure that your provider supports the same requirements of the certificate path algorithm as WebSphere Application Server.

4. Click **OK** and **Save** to save the configuration.
5. Click the name of your certificate store configuration. After you specify the certificate store provider, you must specify either the location of a certificate revocation list or the X.509 certificates. However, you can specify both a certificate revocation list and the X.509 certificates for your certificate store configuration.
6. Under Additional properties, click **Certificate revocation lists**.
7. Click **New** to specify a certificate revocation list path, click **Delete** to delete an existing list reference, or click the name of an existing reference to edit the path. You must specify the fully qualified path to the location where WebSphere Application Server can find your list of certificates that are not valid. For portability reasons, it is recommended that you use the WebSphere Application Server variables to specify a relative path to the certificate revocation lists (CRL). This recommendation is especially important when you are working in a WebSphere Application Server, Network Deployment environment. For example, you might use the *USER_INSTALL_ROOT* variable to define a path such as $*USER_INSTALL_ROOT*/mycertstore/mycrl1. For a list of supported variables, click **Environment** > **WebSphere variables** in the administrative console. The following list provides recommendation for using certificate revocation lists:
   - If CRLs are added to the collection certificate store, add the CRLs for the root certificate authority and each intermediate certificate, if applicable. When the CRL is in the certificate collection store, the certificate revocation status for every certificate in the chain is checked against the CRL of the issuer.
   - When the CRL file is updated, the new CRL does not take effect until you restart the web service application.
   - Before a CRL expires, you must load a new CRL into the certificate collection store to replace the old CRL. An expired CRL in the collection certificate store results in a certificate path (CertPath) build failure.
8. Click **OK** and **Save** to save the configuration.
9. Return to the collection certificate store configuration panel. To access the panel, complete the following steps:
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Manage modules, click ***URI_name***.
   c. Under Web Services Security properties, you can access the key information for the request generator and response generator bindings.
      - For the request generator (sender) binding, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom**.
      - For the response generator (sender) binding, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom**.
   d. Under Additional properties, click **Collection certificate store** > ***certificate_store_name***.
10. Under Additional properties, click **X.509 certificates**.
11. Click **New** to create a X.509 certificate configuration, click **Delete** to delete an existing configuration, or click the name of an existing X.509 certificate configuration to edit its settings. If you are creating a new configuration, enter a name in the Certificate store name field.
12. Specify a path in the X.509 certificate path field. This entry is the absolute path to the location of the X.509 certificate. The collection certificate store is used to validate the certificate path of incoming X.509-formatted security tokens.

    You can use the *USER_INSTALL_ROOT* variable as part of path name. For example, you might type: `USER_INSTALL_ROOT/etc/ws-security/samples/intca2.cer`. Do not use this certificate path for production use. You must obtain your own X.509 certificate from a certificate authority before putting your WebSphere Application Server environment into production.

    Click **Environment** > **WebSphere variables** in the administrative console to configure the *USER_INSTALL_ROOT* variable.

13. Click **OK** and then **Save** to save your configuration.

Results

You have configured the collection certificate store for the generator binding.

What to do next

You must specify a similar collection certificate store configuration for the consumer.

Collection certificate store collection:

Use this page to view a list of certificate stores that contains untrusted, intermediary certificate files awaiting validation. Validation might consist of checking to see if the certificate is on a certificate revocation list (CRL), checking that the certificate is not expired, and checking that the certificate is issued by a trusted signer.

The following list provides recommendations for using CRLs:

- If CRLs are added to the collection certificate store collection, add the CRLs for the root certificate authority and each intermediate certificate, if applicable. When the CRL is in the certificate collection store, the certificate revocation status for every certificate in the chain is checked against the CRL of the issuer.
- When the CRL file is updated, the new CRL does not take effect until you restart the web service application.
- Before a CRL expires, you must load a new CRL into the certificate collection store to replace the old CRL. An expired CRL in the collection certificate store results in a certificate path (CertPath) build failure.

To view the administrative console panel for the collection certificate store on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Collection certificate store**.

To view this administrative console page for the collection certificate store on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access collection certificate stores for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
4. Under Additional properties, you can access collection certificate stores for the following bindings:

- For the Request receiver binding, click **Web services: Server security bindings**. Under Response receiver binding, click **Edit** > **Collection certificate store**.
- For the Response receiver binding, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit** > **Collection certificate store**.

Complete the following steps:

1. Click **New** to specify a new certificate store name and certificate store provider.
2. Click **OK** and messages display at the top of the administrative console panel.
3. Within the messages at the top of the administrative console panel, click **Save**.
4. Return to the collection certificate store collection panel and click **Update runtime** to update the Web Services Security run time with the default binding information, which is found in the `ws-security.xml` file. When you click **Update runtime**, the configuration changes made to the other web services are also updated in the Web Services Security run time.

Certificate store name:

Specifies the name of the certificate store.

Certificate store provider:

Specifies the provider of the certificate store.

Collection certificate store configuration settings:

Use this page to specify the name and the provider for a collection certificate store. A *collection certificate store* is a collection of non-root, certificate authority (CA) certificates and certificate revocation lists (CRLs). This collection of CA certificates and CRLs is used to check the signature of a digitally signed SOAP message.

To view the administrative console panel for the collection certificate store on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Collection certificate store**.
4. Specify a new collection certificate store by clicking **New** or by clicking the collection certificate store name to modify its settings.

To view this administrative console page for the collection certificate store on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access collection certificate stores for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.

- For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Collection certificate store**.
- For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.

4. Under Additional properties, you can access collection certificate stores for the following bindings:
   - For the Request receiver binding click **Edit** > **Collection certificate store**.
   - For the Response receiver binding, click **Edit** > **Collection certificate store**.
5. Specify a new collection certificate store by clicking **New** or by clicking the collection certificate store name to modify its settings.

After configuring a collection certificate store, you can select the new configuration under Certificate store on the token generator and token consumer panels. To access these panels, complete the following steps:

1. Click **Security** > **JAX-WS and JAX-RPC security runtime**.
2. Under JAX-RPC Default Generator Bindings, click **Token generators** or under JAX-RPC Default Consumer Bindings, click **Token consumers**.
3. Click **New** to create a new token generator or token consumer, or click the name of an existing configuration to make modifications.

After you configure your collection certificate store on this panel, you must click **Apply** before configuring either the certificate revocation list or an X.509 certificate. After you configure your certificate revocation list or X.509 certificate, complete the following steps:

1. Click **Save**, at the top of the administrative console panel, which returns you to the list of the configured collection certificate stores.
2. Click **Update runtime** to update the Web Services Security run time with the default binding information, which is found in the `ws-security.xml` file.

Certificate store name:

Specifies the name for the certificate store.

The name of the collection certificate store must be unique in the scope. For example, the name must be unique at the server level. The name specified in **Certificate store name** field is used by other configurations to refer to a pre-defined collection certificate store. For example, the application binding refers to a collection certificate store that is defined on the server level. The application server looks up the collection certificate store based on proximity. For example, if *cert1* is defined as the name of the certificate store on the cell and server levels and *cert1* is referenced in the application binding, the application server uses the server-level collection certificate store.

Certificate Store Provider:

Specifies the provider for the certificate store implementation.

This product supports the IBMCertPath certificate path provider. If you need to use another certificate path provider, define the provider implementation in the provider list within the `java.security` file in the Software Development Kit (SDK).

Data type	String
Default	IBMCertPath

X.509 certificates collection:

Use this page to view a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens.

To view the administrative console page for the collection certificate store on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Collection certificate store**.
4. Click the name of a configured collection certificate store or create a new collection certificate store first.
5. Under Additional properties, click **X.509 certificates**.

To view this administrative console page for an X.509 certificate on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access collection certificate stores for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
4. Under Additional properties, you can access the collection certificate stores for the following bindings.
   - For the Response receiver binding, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit** > **Collection certificate store**.
   - For the Request receiver binding, click **Web services: Server security bindings**. Under Request receiver binding, click **Edit** > **Collection certificate store**.
5. Click the name of a configured collection certificate store or create a new collection certificate store first.
6. Under Additional properties, click **X.509 certificates**.

X.509 certificate path:

Specifies the location of the X.509 certificate.

X.509 certificate configuration settings:

Use this page to specify a list of untrusted, intermediate certificate files. This collection certificate store is used for certificate path validation of incoming X.509-formatted security tokens.

To view the administrative console page for the collection certificate store on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

3. Under Additional properties, click **Collection certificate store**.
4. Click the name of a configured collection certificate store or create a new collection certificate store first.
5. Under Additional properties, click **X.509 certificates**.
6. Specify a new X.509 certificate path by clicking **New** or by clicking the X.509 certificate path to modify its settings.

To view this administrative console page for an X.509 certificate on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access collection certificate stores for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
4. Click the name of a configured collection certificate store or create a new collection certificate store first.
5. Under Additional properties, click **X.509 certificates**.
6. Specify a new X.509 certificate path by clicking **New** or click the X.509 certificate path to modify its settings.

X.509 Certificate Path:

Specifies the absolute path to the location of the X.509 certificate.

As shown in the following example, you can use the USER_INSTALL_ROOT variable as part of the path name: {*USER_INSTALL_ROOT*}/etc/ws-security/samples/intca2.cer. This X.509 certificate path is not for production use. Obtain your own X.509 from a certificate authority before putting your application server environment into production.

You can configure the USER_INSTALL_ROOT variable in the administrative console by clicking **Environment** > **WebSphere Variables**.

Certificate revocation list collection:

Use this page to determine the location of the certificate revocation list (CRL) known to the application server. The Application Server checks the CRL to determine the validity of the client certificate. A certificate that is found in a certificate revocation list might not be expired, but is no longer trusted by the certificate authority (CA) that issued the certificate. The CA might add the certificate to the certificate revocation list if it believes that the client authority is compromised.

View the administrative console panel for the collection certificate store on the server level.

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.

2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Collection certificate store**.
4. Click the name of a configured collection certificate store or create a new collection certificate store first.
5. Under Additional properties, click **Certificate revocation list** > **New**to specify the path to a new list or click the name of the certificate revocation list to modify its path.

View the administrative console page for the collection certificate store on the application level.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access collection certificate stores for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
4. Click the name of a configured collection certificate store or create a new collection certificate store first.
5. Under Additional properties, click **Certificate revocation list** > **New** to specify the path to a new list or click the name of the certificate revocation list to modify its path.
6. Under Additional properties, you can access collection certificate stores for the following bindings:
   - For the Response receiver binding, click **Web services: Client security bindings**. Under Response receiver binding, click **Edit**.
7. Under Additional properties, click **Collection certificate store** > ***certificate_store_name***.
8. Under Additional properties, click **X.509 certificates**.
9. Click **New** and specify the path to the certificate revocation list.

Certificate revocation list path:

Specifies the location where you can find the list of certificates that are not valid.

Certificate revocation list configuration settings:

Use this page to specify a list of certificate revocations that check the validity of a certificate. The application server checks the certificate revocation lists (CRL) to determine the validity of the client certificate. A certificate that is found in a certificate revocation list might not be expired, but is no longer trusted by the certificate authority (CA) that issued the certificate. The CA might add the certificate to the certificate revocation list if it believes that the client authority is compromised.

To view the administrative console panel for the collection certificate store on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

3. Under Additional properties, click **Collection certificate store**.
4. Click the name of a configured collection certificate store or create a new collection certificate store first.
5. Under Additional properties, click **Certificate revocation lists** > **New** to specify the path to a new list or click the name of a certificate revocation list to modify its path.

To view this administrative console page for the collection certificate store on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, you can access collection certificate stores for the following bindings:
   - For the Request generator, click **Web services: Client security bindings**. Under Request generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Request consumer, click **Web services: Server security bindings**. Under Request consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response generator, click **Web services: Server security bindings**. Under Response generator (sender) binding, click **Edit custom** > **Collection certificate store**.
   - For the Response consumer, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom** > **Collection certificate store**.
4. Click the name of a configured collection certificate store or create a new collection certificate store first.
5. Under Additional properties, click **Certificate revocation lists** > **New** to specify the path to a new list or click the name of a certificate revocation list to modify its path.

Certificate revocation list path:

Specifies a fully qualified path to the location where you can find the list of certificates that are not valid.

For portability reasons, it is recommended that you use application server variables to specify a relative path to the certificate revocation list. This recommendation is especially important when you are working in a WebSphere Application Server, Network Deployment environment. For example, you might use the *USER_INSTALL_ROOT* variable to define a path such as *$USER_INSTALL_ROOT/mycertstore/mycrl* where *mycertstore* represents the name of your certificate store and *mycrl* represents the certificate revocation list. For a list of the supported variables, click **Environment** > **WebSphere variables** in the administrative console.

The following list provides recommendations for using CRLs:

- If CRLs are added to the collection certificate store collection, add the CRLs for the root certificate authority and each intermediate certificate, if applicable. When the CRL is in the certificate collection store, the certificate revocation status for every certificate in the chain is checked against the CRL of the issuer.
- When the CRL file is updated, the new CRL does not take effect until you restart the web service application.
- Before a CRL expires, you must load a new CRL into the certificate collection store to replace the old CRL. An expired CRL in the collection certificate store results in a certificate path (CertPath) build failure.

Configuring the collection certificate store for the consumer binding on the application level:

A collection certificate store is a collection of non-root, certificate authority (CA) certificates and certificate revocation lists (CRLs). This collection of CA certificates and CRLs is used to check for a valid signature in a digitally signed SOAP message.

About this task

A collection certificate store is a collection of non-root, certificate authority (CA) certificates and certificate revocation lists (CRLs) that can be used to check for a valid signature in a digitally signed SOAP message. Complete the following steps to configure a collection certificate for the consumer bindings on the application level:

Procedure

1. Locate the collection certificate store configuration panel in the administrative console.
   a. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
   b. Under Modules, click **Manage modules** > ***URI_name***.
   c. Under Web Services Security properties, you can access the collection certificate store information for the response consumer and request consumer bindings.
      - For the response consumer (receiver) binding, click **Web services: Client security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
      - For the request consumer (receiver) binding, click **Web services: Server security bindings**. Under Response consumer (receiver) binding, click **Edit custom**.
   d. Under Additional properties, click **Collection certificate store**.
2. Click **New** to create a collection certificate store configuration, click **Delete** to delete an existing configuration, or click the name of an existing collection certificate store configuration to edit its settings. If you are creating a new configuration, enter a name in the Certificate store name field.

   The name of the collection certificate store must be unique to the level of the application server. For example, if you create the collection certificate store for the application level, the store name must be unique to the application level. The name that is specified in the Certificate store name field is used by other configurations to refer to a predefined collection certificate store. WebSphere Application Server searches for the collection certificate store based on proximity.

   For example, if an application binding refers to a collection certificate store named `cert1`, the Application Server searches for `cert1` at the application level before searching the server level.
3. Specify a certificate store provider in the Certificate store provider field. WebSphere Application Server supports the IBMCertPath certificate store provider. To use another certificate store provider, you must define the provider implementation in the provider list within the *`profile_root`*`/properties/java.security` file. However, make sure that your provider supports the same requirements of the certificate path algorithm as WebSphere Application Server.
4. Click **OK** and **Save** to save the configuration.
5. Click the name of your certificate store configuration. After you specify the certificate store provider, you must specify either the location of a certificate revocation list or the X.509 certificates. However, you can specify both a certificate revocation list and the X.509 certificates for your certificate store configuration.
6. Under Additional properties, click **Certificate revocation lists**.
7. Click **New** to specify a certificate revocation list path, click **Delete** to delete an existing list reference, or click the name of an existing reference to edit the path. You must specify the fully qualified path to the location where WebSphere Application Server can find your list of certificates that are not valid. For portability reasons, it is recommended that you use the WebSphere Application Server variables to specify a relative path to the certificate revocation lists (CRL). This recommendation is especially important when you are working in a WebSphere Application Server, Network Deployment environment. For example, you might use the *USER_INSTALL_ROOT* variable to define a path such

as $*USER_INSTALL_ROOT*/mycertstore/mycrl1. For a list of supported variables, click **Environment** > **WebSphere variables** in the administrative console. The following list provides recommendation for using certificate revocation lists:

- If CRLs are added to the collection certificate store, add the CRLs for the root certificate authority and each intermediate certificate, if applicable. When the CRL is in the certificate collection store, the certificate revocation status for every certificate in the chain is checked against the CRL of the issuer.
- When the CRL file is updated, the new CRL does not take effect until you restart the web service application.
- Before a CRL expires, you must load a new CRL into the certificate collection store to replace the old CRL. An expired CRL in the collection certificate store results in a certificate path (CertPath) build failure.

8. Click **OK** and **Save** to save the configuration.
9. Return to the Collection certificate store configuration panel. See the first few steps of this article to locate the collection certificate store panel.
10. Under Additional properties, click **X.509 certificates**.
11. Click **New** to create a new configuration for X.509 certificates, click **Delete** to delete an existing configuration, or click the name of an existing X.509 certificate configuration to edit its settings. If you are creating a new configuration, enter a name in the Certificate store name field.
12. Specify a path in the X.509 certificate path field. This entry is the absolute path to the location of the X.509 certificates. The collection certificate store is used to validate the certificate path of incoming X.509-formatted security tokens.

    You can use the *USER_INSTALL_ROOT* variable as part of the path name. For example, you might type: `USER_INSTALL_ROOT/etc/ws-security/samples/intca2.cer`. Do not use this certificate path for production use. You must obtain your own X.509 certificate from a certificate authority before putting your WebSphere Application Server environment into production.

    Click **Environment** > **WebSphere variables** in the administrative console to configure the *USER_INSTALL_ROOT* variable.
13. Click **OK** and then **Save** to save your configuration.

Results

You have configured the collection certificate store for the consumer binding.

What to do next

You must configure a token consumer configuration that references this certificate store configuration.

Configuring the collection certificate on the server level:

Collection certificate stores contain untrusted, intermediary certificate files awaiting validation. You can configure the collection certificate store on the server level and the cell level.

About this task

Validation might consist of checking for a valid signature in a digitally signed SOAP message to see if the certificate is on a certificate revocation list (CRLs), checking that the certificate is not expired, and checking that the certificate is issued by a trusted signer.

Complete the following steps to configure a collection certificate store on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
2. Under Additional properties, click **Collection certificate store**.
3. Click **New** to create a collection certificate store configuration, click **Delete** to delete an existing configuration, or click the name of an existing collection certificate store configuration to edit its settings. If you are creating a new configuration, enter a name in the Certificate store name field. For example, you might name your certificate store `sig_certstore`.

   The name of the collection certificate store must be unique to the level of the application server. For example, if you create the collection certificate store for the server level, the store name must be unique to the server level. The name that is specified in the Certificate store name field is used by other configurations to refer to a predefined collection certificate store. WebSphere Application Server searches for the collection certificate store based on proximity.

   For example, if an application binding refers to a collection certificate store named `cert1`, the Application Server searches for `cert1` at the application level before searching the server level.
4. Specify a certificate store provider in the Certificate store provider field. WebSphere Application Server supports the IBMCertPath certificate store provider. To use another certificate store provider, you must define the provider implementation in the provider list within the *`profile_root`*`/properties/java.security` file. However, make sure that your provider supports the same requirements of the certificate path algorithm as WebSphere Application Server.
5. Click **OK** and **Save** to save the configuration.
6. Click the name of your certificate store configuration. After you specify the certificate store provider, you must specify either the location of a certificate revocation list or the X.509 certificates. However, you can specify both a certificate revocation list and the X.509 certificates for your certificate store configuration.
7. Under Additional properties, click **Certificate revocation lists**. For the generator binding, a certificate revocation list (CRL) is used when it is included in a generated security token. For example, a security token might be wrapped in a PKCS#7 format with a CRL. For more information on certificate revocation lists, see Certificate revocation list.
8. Click **New** to specify a certificate revocation list path, click **Delete** to delete an existing list reference, or click the name of an existing reference to edit the path. You must specify the fully qualified path to the location where WebSphere Application Server can find your list of certificates that are not valid. WebSphere Application Server uses the certificate revocation list to check the validity of the sender certificate.

   For portability reasons, it is recommended that you use the WebSphere Application Server variables to specify a relative path to the certificate revocation lists. This recommendation is especially important when you are working in a WebSphere Application Server, Network Deployment environment.

   For example, you might use the *USER_INSTALL_ROOT* variable to define a path such as *$USER_INSTALL_ROOT/mycertstore/mycrl1* where *mycertstore* represents the name of your certificate store and *mycrl1* represents the certificate revocation list. For a list of supported variables, click **Environment** > **WebSphere variables** in the administrative console. The following list provides recommendations for using certificate revocation lists:

   - If CRLs are added to the collection certificate store, add the CRLs for the root certificate authority and each intermediate certificate, if applicable. When the CRL is in the certificate collection store, the certificate revocation status for every certificate in the chain is checked against the CRL of the issuer.

- When the CRL file is updated, the new CRL does not take effect until you restart the web service application.
- Before a CRL expires, you must load a new CRL into the certificate collection store to replace the old CRL. An expired CRL in the collection certificate store results in a certificate path (CertPath) build failure.

9. Click **OK** and then **Save** to save the configuration.
10. Return to the Collection certificate store configuration panel.
11. Under Additional properties, click **X.509 certificates**. The X.509 certificate configuration specifies intermediate certificate files that are used for certificate path validation of incoming X.509-formatted security tokens.
12. Click **New** to create an X.509 certificate configuration, click **Delete** to delete an existing configuration, or click the name of an existing X.509 certificate configuration to edit its settings. If you are creating a new configuration, enter a name in the Certificate store name field.
13. Specify a path in the X.509 certificate path field. This entry is the absolute path to the location of the X.509 certificate. The collection certificate store is used to validate the certificate path of the incoming X.509-formatted security tokens.

    You can use the *USER_INSTALL_ROOT* variable as part of path name. For example, you might type: `$USER_INSTALL_ROOT/etc/ws-security/samples/intca2.cer`. Do not use this certificate path for production use. You must obtain your own X.509 certificate from a certificate authority before putting your WebSphere Application Server environment into production.

    Click **Environment** > **WebSphere variables** in the administrative console to configure the *USER_INSTALL_ROOT* variable.
14. Click **OK** and then **Save** to save your configuration.
15. Return to the Collection certificate store collection panel and click **Update run time** to update the Web Services Security run time with the default binding information, which is located in the `ws-security.xml` file. When you click **Update run time**, the configuration changes made to other web services are also updated in the run time for Web Services Security. Policy sets can only be used with JAX-WS applications. Policy sets cannot be used for JAX-RPC applications.

Results

You have configured the collection certificate store for the server level.

Configuring trusted ID evaluators on the server level:

You can configure trusted identity (ID) evaluators. The trusted ID evaluator determines whether or not to trust the identity-asserting authority.

About this task

This task provides the steps that are needed to configure trusted identity (ID) evaluators. The trusted ID evaluator determines whether to trust the identity-asserting authority. After the ID is trusted, the WebSphere Application Server issues the proper credentials based on the identity, which are used in a downstream call to another server for invoking resources. The trusted ID evaluator implements the `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluator` interface.

Complete the following steps to configure the trusted ID evaluators on the server level:

Procedure

1. Access the default bindings for the server level.
   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

2. Under Additional properties, click **Trusted ID evaluators**.
3. Click **New** to create a trusted ID evaluator configuration, click **Delete** to delete an existing configuration, or click the name of an existing configuration to edit the settings. If you are creating a new configuration, enter a unique name for the trusted ID evaluator configuration in the Trusted ID evaluator name field. This field specifies the name that is used by the application binding to refer to a trusted identity (ID) evaluator that is defined in the default binding.
4. Specify a class name in the Trusted ID evaluator class name field. The default class name is `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluatorImpl`. The specified trusted ID evaluator class name must implement the `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluator` class. When you use the default `TrustedIDEvaluator` class, you must specify the name and value properties for the default trusted ID evaluator to create the trusted ID list for evaluation.
5. Under Additional properties, click **Properties > New**.
6. Specify the trusted ID evaluator name as a property name. You must specify the trusted ID evaluator name in the form, `trustedId_n`, where `_n` is an integer from zero (0) to n.
7. Specify the trusted ID as a property value.

```
property name="trustedId_0", value="CN=Bob,O=ACME,C=US"
property name="trustedId_1, value="user1"
```

   If a distinguished name (DN) is used, the space is removed for comparison.
8. Click **OK** and then **Save**.

Results

You have configured the trusted ID evaluators at the server level.

Trusted ID evaluator collection:

Use this page to view a list of trusted identity (ID) evaluators. The trusted ID evaluator determines whether to trust the identity-asserting authority. After the ID is trusted, the application server issues the proper credentials based on the identity, which are used in a downstream call for invoking resources. The trusted ID evaluator implements the `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluator` interface.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To view this administrative console page for trusted ID evaluators on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Trusted ID evaluators**.
4. Click **New** to create a trusted ID evaluator or click **Delete** to a delete a trusted ID evaluator.

To view this administrative console page for trusted ID evaluators on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage Modules** > ***URI_name***.
3. Under Web Services Security Properties, click **Web services: Server security bindings**.
4. Under Request receiver binding, click **Edit**.

5. Click **Trusted ID evaluators**.
6. Click **New** to create a trusted ID evaluator or click **Delete** to delete a trusted ID evaluator.

Important: Trusted ID evaluators are only required for the request consumer (Version 6.*x* applications), if identity assertion is configured.

Using this trusted ID evaluator collection panel, complete the following steps:

1. Specify a trusted ID evaluator name and a trusted ID evaluator class name.
2. Save your changes by clicking **Save** in the messages section at the top of the administrative console.
3. Click **Update run time** to update the Web Services Security run time with the default binding information, which is found in the `ws-security.xml` file. The configuration changes made to the other web services also are updated in the Web Services Security run time.

Trusted ID evaluator name:

Specifies the unique name of the trusted ID evaluator.

Trusted ID evaluator class name:

Specifies the class name of the trusted ID evaluator.

Trusted ID evaluator configuration settings:

Use this information to configure trust identity (ID) evaluators.

This administrative console page applies only to Java API for XML-based RPC (JAX-RPC) applications.

To view this administrative console page for trusted ID evaluators on the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Trusted ID evaluators**.
4. Click **New** to create a trusted ID evaluator or click the name of an existing configuration to modify the settings.

To view this administrative console page for trusted ID evaluators on the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, click **Web services: Server security bindings**.
4. Under Request receiver binding, click **Edit**.
5. Click **Trusted ID evaluators**.
6. Click **New** to create a trusted ID evaluator or click **Delete** to delete a trusted ID evaluator.

Important: Trusted ID evaluators are only required for the request consumer (Version 6.*x* applications), if identity assertion is configured.

You can specify one of the following options:

None Choose this option if you are not specifying a trusted ID evaluator.

Existing evaluator definition
Choose this option to specify a currently defined trusted ID evaluator.

Binding evaluator definition
Choose this option to specify a new trusted ID evaluator. A description of the required fields follows.

Trusted ID evaluator name:

Specifies the name that is used by the application binding to refer to a trusted identity (ID) evaluator that is defined in the default binding.

Trusted ID evaluator class name:

Specifies the class name of the trusted ID evaluator.

The specified trusted ID evaluator class name must implement the `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluator` interface. The default TrustedIDEvaluator class is `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluatorImpl`. When you use this default TrustedIDEvaluator class, you must specify the name and the value properties for the default trusted ID evaluator to create the trusted ID list for evaluation.

To specify the name and value properties, complete the following steps:

1. Under Additional properties, click **Properties** > **New**.
2. Specify the trusted ID evaluator name as a property name. You must specify the trusted ID evaluator name in the form, `trustedId_n`, where `_n` is an integer from zero (0) to n.
3. Specify the trusted ID as a property value.

For example:

```
property name="trustedId_0", value="CN=Bob,O=ACME,C=US"
property name="trustedId_1", value="user1"
```

If a distinguished name (DN) is used, the space is removed for comparison.

Default `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluatorImpl`

See the programming model information in the documentation for an explanation of how to implement the `com.ibm.wsspi.wssecurity.id.TrustedIDEvaluator` interface.

rrdSecurity.props file:

Remote request dispatcher (RRD) supports LTPA and security attribute propagation for Web Services Security (WS-Security). You can enable token propagation in the `<was_install>/profiles/<profileName>/properties/rrdSecurity.props` file.

The `rrdSecurity.props` file contains comments to describe the security attributes.

The following is the format of the `rrdSecurity.props` file. The default values are in bold face type.

- `LTPAPropagation= ( True | False)`
- `SecurityAttributePropagation= ( True | False)`
- `SSLRequired= ( True | False)`

The WS-Security run time inspects the run as (invocation) subject and propagates the security tokens in the subject. The default setting is to only propagate the LTPA tokens.

Custom security tokens can be passed as attributes of the LTPA tokens. The security attribute propagation support uses the same pluggable JAAS login module as the CSIv2 support. The security attribute is not signed or encrypted, therefore, you should not send the attribute in clear text form. You must require SSL to ensure integrity and confidentiality. If SSL is not required, RRD uses the same scheme, such as HTTP or HTTPS, to make the web services call that the original request used.

You must also configure the target web service to validate the LTPA tokens and security attributes.

Enabling or disabling single sign-on interoperability mode for the LTPA token

You can set an interoperability flag on the token generator to determine whether an LTPA Version 1 token or an LTPA Version 2 token is retrieved when a request message is received.

About this task

In WebSphere Application Server Version 7.0 and later, a flag is set in the global security settings to enable single sign-on interoperability mode for the LTPA token. This option determines whether an LTPA Version 1 token or an LTPA Version 2 token is sent when a message request is received. When the interoperability flag is set to true, then the AuthenticationToken is an LTPA Version 1 token, and the SingleSignonToken is an LTPA Version 2 token. When the interoperability flag is set to false, then both the AuthenticationToken and the SingleSignonToken are LTPA Version 2 tokens.

When the interoperability mode is enabled (the flag is set to true), and the Web Services Security binding configuration specifies LTPA Version 1 as the token, the AuthenticationToken is used to retrieve the token that is sent with the message. If interoperability mode is not enabled (the flag is set to false), and the Web Services Security binding configuration specifies LTPA Version 1 as the token, an exception error is logged.

You can disable the interoperability checking function by setting the custom property, com.ibm.wsspi.wssecurity.tokenGenerator.ltpav1.pre.v7, on the token generator. This setting determines the LTPA token without checking the state of the interoperability flag, providing compatibility with servers running WebSphere Application Server Version 6.1 and earlier.

To enforce use of the LTPA Version 2 token, edit the token settings, and set the **Enforce token version** option for the token.

Procedure

1. Click **Applications > Application Types > WebSphere enterprise applications**.
2. Select an application that contains web services. The application must contain a service provider or a service client.
3. Click the **Service provider policy sets and bindings** link or the **Service client policy sets and bindings** link in the Web Services Properties section.
4. Select a binding. You must have previously attached a policy set and assigned an application specific binding.
5. Click the **WS-Security** policy in the Policies table.
6. Click the **Authentication and protection** link in the Main message security policy bindings section.
7. Click a consumer or generator token link from the Protection Tokens table.
8. Select the **Enforce token version** check box after the **Token type** field.

Enabling cryptographic keys stored in hardware devices for Web Services Security

You can enable Web Services Security by using cryptographic hardware devices for both web service clients and web service providers that are running in the WebSphere® Application Server environment.

Enabling hardware cryptographic devices for Web Services Security

You can enable Web Services Security by using cryptographic hardware devices for both web service clients and web service providers that are running in the WebSphere Application Server environment. A cryptographic token is a hardware or software device with a built-in keystore implementation. Cryptographic devices are used to manage certificates stored on the cryptographic tokens. These devices are also called *smartcards*. You enable hardware cryptographic devices for Web Service Security by either using keys that are stored in hardware devices or by using keys stored in a Java keystore file.

About this task

Web Services Security using cryptographic hardware devices is supported for both web (JavaServer Pages (JSP) or servlet) and Enterprise JavaBeans (EJB) web service clients. You can enable Web Services Security by using cryptographic hardware devices for both web service clients and web service providers that are running in the WebSphere Application Server environment.

There are two ways to enable hardware cryptographic devices for Web Service Security: use keys that are stored in hardware devices or use keys stored in a Java keystore file.

Procedure

1. Determine whether to use keys that are stored in hardware devices or in a Java keystore file for the individual application.
2. Enable hardware cryptographic devices for Web Service Security by using one of the following two methods:
   - Enable cryptographic operations on hardware devices. See “Configuring hardware cryptographic devices for Web Services Security” for more details.
   - Enable cryptographic keys that are stored in hardware devices. See “Enabling cryptographic keys stored in hardware devices in Web Services Security” on page 3441

   Note: Hardware cryptographic devices for Web Services Security are not supported on the Java Platform, Enterprise Edition (Java EE) Application Client on distributed platform.

Configuring hardware cryptographic devices for Web Services Security:

Before you can use a hardware cryptographic device, you must configure and enable it. You must first configure a hardware cryptographic device using the Secure Sockets Layer (SSL) certificate and key management panels in the administrative console. The key for the cryptographic operation can be stored in an ordinary Java keystore file and need not be stored on the hardware devices.

Before you begin

You must first configure a hardware cryptographic device using the Secure Sockets Layer (SSL) certificate and key management panels in the administrative console.

Note: Fix packs that include updates to the Software Development Kit (SDK) might overwrite unrestricted policy files. Back up unrestricted policy files before you apply a fix pack and reapply these files after the fix pack is applied.

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** and then select the server name.
2. Under **Security**, select **JAX-WS and JAX-RPC security runtime**.
3. Under Cryptographic Hardware, select **Enable cryptographic operations on hardware device** and then specify the name of the hardware cryptographic device configuration name. For more information, read about configuring a hardware cryptographic keystore.

4. Click **OK**.

Results

This procedure configures a hardware cryptographic device for all Web Services Security applications running on this application server.

Enabling cryptographic keys stored in hardware devices in Web Services Security:

You can enable individual web service applications to use cryptographic keys stored in hardware devices in Web Services Security.

Before you begin

You must first configure the hardware acceleration device using the key management panels in the administrative console. See “Configuring hardware cryptographic devices for Web Services Security” on page 3440

Procedure

1. In the administrative console, click **Servers** > **Server types** > **WebSphere application servers** and then select the server name.
2. Under **Security**, click **JAX-WS and JAX-RPC security runtime**.
3. Under **Additional properties**, click **key locators**.
4. Select the key locator name.
5. Under **Key store**, specify the name of the keystore configuration.

   If the keystore reference is specified to a hardware device configuration, the Web Services Security runtime first attempts to obtain the cryptographic algorithm from the hardware device. If the hardware device is not supported or if it fails, the runtime for Web Services Security obtains the cryptographic algorithm from the security providers list. Read about creating a keystore configuration for a preexisting keystore file for more information about how to create the name of a keystore configuration.
6. Click **OK**.

Results

If the name of the keystore reference is a Java keystore file, a hardware acceleration device that is configured at the application server level (`ws-security.xml`) will be used for cryptographic operations.

Configuring XML digital signature for Version 5.x web services with the administrative console

XML digital signature provides both message integrity and authentication capabilities when it is used with SOAP messages. XML digital signature is one of the methods WebSphere® Application Server provides to secure web services. You can use the WebSphere® Application Server administrative console to configure XML digital signature.

Login mappings collection

Use this page to view a list of configurations for validating security tokens within incoming messages. Login mappings map an authentication method to a Java Authentication and Authorization Service (JAAS) login configuration to validate the security token. Four authentication methods are predefined in the WebSphere Application Server: BasicAuth, Signature, IDAssertion, and Lightweight Third Party Authentication (LTPA).

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with

WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications. Version 5.*x* applications are based on Java 2 platform, Enterprise Edition (J2EE) 1.3.

To view this administrative console page for the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Login mappings**.
4. Click either **New** to create a new login mapping configuration or click the name of an existing configuration.

To view this administrative console page for the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security properties, click **Web services: Server security bindings**.
4. Click **Edit** under Request receiver binding.
5. Click **Login mappings**.

If you click **Update runtime**, the Web Services Security run time is updated with the default binding information, which is contained in the `ws-security.xml` file that was previously saved. After you specify the authentication method, the JAAS configuration name, and the Callback Handler Factory class name on this panel, you must complete the following steps:

1. Click **Save** in the messages section at the top of the administrative console.
2. Click **Update runtime**. When you click **Update runtime**, the configuration changes made to the other web services also are updated in the Web Services Security run time.

Important: If the login mapping configuration is not found on the application level, the web services run time searches for the login mapping configuration on the server level.

Authentication method:

Specifies the authentication method used for validating the security tokens.

The following authentication methods are available:

BasicAuth
The basic authentication method includes both a user name and a password in the security token. The information in the token is authenticated by the receiving server and is used to create a credential.

Signature
The signature authentication method sends an X.509 certificate as a security token. For Lightweight Directory Access Protocol (LDAP) registries, the distinguished name (DN) is mapped to a credential, which is based on the LDAP certificate filter settings. For local OS registries, the first attribute of the certificate, usually the common name (CN) is mapped directly to a user name in the registry.

IDAssertion
The identity assertion method maps a trusted identity (ID) to a WebSphere Application Server credential. This authentication method only includes a user name in the security token. An additional token is included in the message for trust purposes. When the additional token is trusted, the IDAssertion token user name is mapped to a credential.

LTPA Lightweight Third Party Authentication (LTPA) validates an LTPA token.

JAAS configuration name:

Specifies the name of the Java Authentication and Authorization Service (JAAS) configuration.

Callback handler factory class name:

Specifies the name of the factory for the CallbackHandler class.

Login mapping configuration settings

Use this page to specify the Java Authentication and Authorization Service (JAAS) login configuration settings that are used to validate security tokens within incoming messages.

Important: There is an important distinction between Version 6 and later applications. The information in this article supports Version 6.*x* applications only that are used with WebSphere Application Server Version 6.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

To view this administrative console page for the server level, complete the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
3. Under Additional properties, click **Login mappings**.
4. Click either **New** to create a new login mapping configuration or click the name of an existing configuration.

To use this administrative console page for the application level, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_name***.
3. Under Web Services Security Properties, click **Web services: Server security bindings**.
4. Click **Edit** under Request receiver binding.
5. Click **Login mappings**.
6. Click either **New** to create a new login mapping configuration or click the name of an existing configuration.

Important: If the login mapping configuration is not found on the application level, the web services run time searches for the login mapping configuration on the server level.

Authentication method:

Specifies the method of authentication.

You can use any string, but the string must match the element in the service-level configuration. The following words are reserved and have special meanings:

BasicAuth
Uses both a user name and a password.

IDAssertion
Uses only a user name, but requires that additional trust is established on the receiving server using a TrustedIDEvaluator mechanism.

Signature
Uses the distinguished name (DN) of the signer.

LTPA Validates a token.

JAAS configuration name:

Specifies the name of the Java Authentication and Authorization Service (JAAS) configuration.

Among the predefined system login configurations that you can use are the following:

system.wssecurity.IDAssertion
Enables a version 6.*x* application to use identity assertion to map a user name to a WebSphere Application Server credential principal.

system.wssecurity.Signature
Enables a version 6.*x* application to map a distinguished name (DN) in a signed certificate to a WebSphere Application Server credential principal.

system.LTPA_WEB
Processes login requests that are used by the web container such as servlets and JavaServer Pages (JSP) files.

system.WEB_INBOUND
Handles logins for web application requests, which include servlets and JavaServer Pages..

system.RMI_INBOUND
Handles logins for inbound Remote Method Invocation (RMI) requests.

system.DEFAULT
Handles the logins for inbound requests made by internal authentications and most of the other protocols except web applications and RMI requests.

system.RMI_OUTBOUND
Processes RMI requests that are sent outbound to another server when the com.ibm.CSIOutboundPropagationEnabled property is `true`. This property is set in the CSIv2 authentication panel. To access the panel, click **Security** > **Global security**. Expand RMI/IIOP security, then click on **CSIv2 Outbound authentication**. To set the `com.ibm.CSIOutboundPropagationEnabled` property, select **Security attribute propagation**.

system.wssecurity.X509BST
Verifies an X.509 binary security token (BST) by checking the validity of the certificate and the certificate path.

system.wssecurity.PKCS7
Verifies an X.509 certificate with a certificate revocation list in a PKCS7 object.

system.wssecurity.PkiPath
Verifies an X.509 certificate with a public key infrastructure (PKI) path.

system.wssecurity.UsernameToken
Verifies basic authentication (user name and password).

These system login configurations are defined on the System logins panel, which is accessible by completing the following steps:

1. Click **Security** > **Global security**.
2. Expand Java Authentication and Authorization Service, then click **System logins**.

Attention: The predefined system login configurations are listed on the System logins configuration panel without the system prefix. For example, the system.wssecurity.UsernameToken configuration listed in the Java Authentication and Authorization Service (JAAS) configuration name option corresponds to the wssecurity.UsernameToken configuration that is on the System logins configuration panel.

You can use the following predefined application login configurations:

ClientContainer
Specifies the login configuration that is used by the client container application, which uses the CallbackHandler API that is defined in the deployment descriptor of the client container.

WSLogin
Specifies whether all applications can use the WSLogin configuration to perform authentication for the WebSphere Application Server security run time.

DefaultPrincipalMapping
Specifies the login configuration used by Java 2 Connectors (J2C) to map users to principals that are defined in the J2C authentication data entries.

These application login configurations are defined on the Application logins panel, which is accessible by completing the following steps:

1. Click **Security** > **Global security**.
2. Expand Java Authentication and Authorization Service, then click **Application logins**.

Do not remove these predefined system or application login configurations. Within these configurations, you can add module class names and specify the order in which WebSphere Application Server loads each module.

Callback handler factory class name:

Specifies the name of the factory for the CallbackHandler class.

You must implement the `com.ibm.wsspi.wssecurity.auth.callback.CallbackHandlerFactory` class in this field.

Token type URI:

Specifies the namespace Uniform Resource Identifiers (URI), which denotes the type of security token that is accepted.

If binary security tokens are accepted, the value denotes the ValueType attribute in the element. The ValueType element identifies the type of security token and its namespace. If Extensible Markup Language (XML) tokens are accepted, the value denotes the top-level element name of the XML token.

If the reserved words are specified previously in the Authentication method field, this field is ignored.

Data type: Unicode characters except for non-ASCII characters, but including the number sign (#), the percent sign (%), and the square brackets ([ ]).

Token type local name:

Specifies the local name of the security token type, for example, X509v3.

If binary security tokens are accepted, the value denotes the ValueType attribute in the element. The ValueType attribute identifies the type of security token and its namespace. If Extensible Markup Language (XML) tokens are accepted, the value denotes the top-level element name of the XML token.

If the reserved words are specified previously in the Authentication method field, this field is ignored.

Nonce maximum age:

Specifies the time, in seconds, before the nonce timestamp expires. Nonce is a randomly generated value.

You must specify a minimum of 300 seconds for the Nonce maximum age field. However, the maximum value cannot exceed the number of seconds specified in the Nonce cache timeout field for the server level.

You can specify the Nonce maximum age value for the server level by completing the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

Important: The Nonce maximum age field on this panel is optional and only valid if the BasicAuth authentication method is specified. If you specify another authentication method and attempt to specify values for this field, the following error message displays and you must remove the specified value: `Nonce is not supported for authentication methods other than BasicAuth.`

If you specify the BasicAuth method, but do not specify values for the Nonce maximum age field, the Web Services Security run time searches for a Nonce maximum age value on the server level.

Default	300 seconds
Range	300 to Nonce cache timeout seconds

Nonce clock skew:

Specifies the clock skew value, in seconds, to consider when WebSphere Application Server checks the freshness of the message. Nonce is a randomly generated value.

You can specify the **Nonce clock skew** value for the server level by completing the following steps:

1. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
2. Under Security, click **JAX-WS and JAX-RPC security runtime**.

Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

You must specify a minimum of zero (0) seconds for the Nonce Clock Skew field. However, the maximum value cannot exceed the number of seconds that is specified in the Nonce maximum age field on this Login mappings panel.

Important: The Nonce clock skew field on this panel is optional and only valid if the BasicAuth authentication method is specified. If you specify another authentication method and attempt to specify values for this field, the following error message displays and you must remove the specified value: `Nonce is not supported for authentication methods other than BasicAuth.`

Note: If you specify BasicAuth, but do not specify values for the Nonce clock skew field, WebSphere Application Server searches for a Nonce clock skew value on the server level.

Default	0 seconds
Range	0 to Nonce Maximum Age seconds

Configuring nonce using Web Services Security tokens

Nonce is a randomly generated, cryptographic token that is used to thwart the highjacking of user name tokens, which are used with SOAP messages. Use nonce in conjunction with the BasicAuth authentication method.

About this task

Important: The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

You can configure nonce at the application level and server level.

If you configure nonce on the application level and the server level, the values specified for the application level take precedence over the values specified for the server level.

You must consider the order of precedence:

1. Application level
2. Server level

Complete these high-level tasks in the order listed:

Procedure

1. Configure nonce for the application level.
2. Configure nonce for the server level.

What to do next

After completing these steps, restart the server if it has not already been restarted.

Configuring nonce for the server level:

Nonce is a randomly generated, cryptographic token that is used to prevent the theft of username tokens, which are used with SOAP messages. Nonce is used in conjunction with the basic authentication (BasicAuth) method. You can configure nonce for the server level by using the WebSphere Application Server administrative console.

About this task

Important: The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

You can configure nonce at the application level and the cell level.

However, you must consider the order of precedence:

1. Application level
2. Server level

If you configure nonce on the application level and the server level, the values specified for the application level take precedence over the values specified for the server level.

In a WebSphere Application Server (base) or WebSphere Application Server, Express environment, you must specify values for the Nonce cache timeout, Nonce maximum age, and Nonce clock skew fields on the server level to use nonce effectively.

Complete the following steps to configure nonce on the server level:

Procedure

1. Connect to the administrative console.

   Type `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.
2. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_level***.
3. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
4. Specify a value, in seconds, for the Nonce cache timeout field. The value specified for the Nonce cache timeout field indicates how long the nonce remains cached before it is expunged. You must specify a minimum of 300 seconds. However, if you do not specify a value, the default is 600 seconds. This field is required for the server level.
5. Specify (optional) a value, in seconds, for the Nonce maximum age field.

   The value specified for the Nonce Maximum Age field indicates how long the nonce is valid. You must specify a minimum of 300 seconds, but the value cannot exceed the number of seconds specified for the Nonce cache timeout field on the server level.

   This field is required for the server level.
6. Specify a value, in seconds, for the Nonce clock skew field. The value specified for the Nonce clock skew field specifies the amount of time, in seconds, to consider when the message receiver checks the timeliness of the value. Consider the following information when you set this value:
   - Difference in time between the message sender and the message receiver if the clocks are not synchronized.
   - Time needed to encrypt and transmit the message.
   - Time needed to get through network congestion.

   You must specify at least 0 seconds for the Nonce clock skew field. However, the maximum value cannot exceed the number of seconds specified in the Nonce maximum age field on the server level. If you do not specify a value, the default is 0 seconds.
7. Restart the server. If you change the Nonce cache timeout value and do not restart the server, the change is not recognized by the server.

Configuring nonce for the application level:

Nonce is a randomly generated, cryptographic token that is used to thwart the highjacking of Username tokens, which are used with SOAP messages. Use nonce in conjunction with the basic authentication (BasicAuth) method. You can configure nonce for the application level by using the WebSphere Application Server administrative console.

About this task

Important: The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

You can configure nonce at the application level and server level.

However, you must consider the order of precedence:

1. Application level
2. Server level

If you configure nonce on the application level and the server level, the values specified for the application level take precedence over the values specified for the server level.

Procedure

1. Connect to the administrative console.

   Type `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.
2. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
3. Under Manage modules, click ***URI_name***.
4. Under Web Services Security Properties, click **Web services: Server security bindings**.
5. Click **Edit** under Request receiver binding
6. Under Additional properties, click **Login mappings** > **New**.
7. Specify (optional) a value, in seconds, for the **Nonce maximum age** field. This panel is optional and only valid if the BasicAuth authentication method is specified. If you specify another authentication method and attempt to specify values for this field, the following error message displays and you must remove the specified value:

```
Nonce is not supported for authentication methods other than
BasicAuth.
```

   If you specify BasicAuth, but do not specify values for the **Nonce maximum age** field, the Web Services Security runtime searches for a nonce maximum age value on the server level.

   The value specified for the **Nonce maximum age** field indicates how long the nonce is valid. You must specify a minimum of 300 seconds; however, the value cannot exceed the number of seconds that is specified for the **Nonce cache timeout** field for the server level.

   You can specify the nonce cache timeout value for the server level by completing the following steps:

   a. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
   b. Under Security, click **JAX-WS and JAX-RPC security runtime**.

      Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
8. Specify (optional) a value, in seconds, for the **Nonce clock skew** field. The value specified for the **Nonce clock skew** field specifies the amount of time, in seconds, to consider when the message receiver checks the timeliness of the value. This panel is optional and only valid if the BasicAuth authentication method is specified. If you specify another authentication method and attempt to specify values for this field, the following error message displays and you must remove the specified value:

```
Nonce is not supported for authentication methods other than
BasicAuth.
```

   If you specify BasicAuth, but do not specify values for the **Nonce clock skew** field, the Web Services Security runtime searches for a Nonce clock skew value on the server level.

   Consider the following information when you set this value:

   - Difference in time between the message sender and the message receiver if the clocks are not synchronized.
   - Time needed to encrypt and transmit the message.
   - Time needed to get through network congestion.
9. Restart the server.

Configuring trust anchors using the administrative console

Use the WebSphere Application Server administrative console to configure trust anchors that specify key stores which contain trusted root certificates to validate the signer certificate.

Before you begin

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

This document describes how to configure trust anchors or trust stores at the application level. It does not describe how to configure trust anchors at the server or cell level. Trust anchors defined at the application level have a higher precedence over trust anchors defined at the server or cell level. For more information on creating and configuring trust anchors at the server or cell level, see either Configuring the server security bindings using an assembly tool or “Configuring the server security bindings using the administrative console” on page 3457.

You can configure an application-level trust anchor using an assembly tool or the administrative console. This document describes how to configure the application-level trust anchor using the administrative console.

About this task

A trust anchor specifies key stores that contain trusted root certificates, which validate the signer certificate. These key stores are used by the request receiver (as defined in the `ibm-webservices-bnd.xmi` file) and the response receiver (as defined in the `ibm-webservicesclient-bnd.xmi` file when web services are acting as client) to validate the signer certificate of the digital signature. The keystores are critical to the integrity of the digital signature validation. If they are tampered with, the result of the digital signature verification is doubtful and comprised. Therefore, it is recommended that you secure these keystores. The binding configuration specified for the request receiver in the `ibm-webservices-bnd.xmi` file must match the binding configuration for the response receiver in the `ibm-webservicesclient-bnd.xmi` file.

The following steps are for the client-side response receiver, which is defined in the `ibm-webservicesclient-bnd.xmi` file and the server-side request receiver, which is defined in the `ibm-webservices-bnd.xmi` file.

Procedure

1. Configure an assembly tool to work with a Java Platform, Enterprise Edition (Java EE) enterprise application. For more information, see the related information on Assembly Tools.
2. Create a web services-enabled Java EE enterprise application. See either Configuring the server security bindings using an assembly tool or “Configuring the server security bindings using the administrative console” on page 3457 for an introduction on how to manage Web Services Security binding information on the server.
3. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***enterprise_application***.
4. Under Manage modules, click ***URI_name***.
5. Under Web Services Security Properties, click **Web services: client security bindings** to edit the response receiver binding information, if web services are acting as a client.
   a. Under Response receiver binding, click **Edit**.
   b. Under Additional properties, click **Trust anchors**.
   c. Click **New** to create a new trust anchor.
   d. Enter a unique name within the request receiver binding for the Trust anchor name field. The name is used to reference the trust anchor that is defined.

e. Enter the key store password, path, and key store type.
f. Click the trust anchor name link to edit the selected trust anchor.
g. Click **Remove** to remove the selected trust anchor or anchors.

When you start the application, the configuration is validated in the run time while the binding information is loading.

6. Return to the web services-enabled module panel accessed in step 2.
7. Under Web Services Security Properties, click **Web services: server security bindings** to edit the request receiver binding information.
   a. Under Request receiver binding, click **Edit**.
   b. Under Additional properties, click **Trust anchors**.
   c. Click **New** to create a new trust anchor

   Enter a unique name within the request receiver binding for the Trust anchor name field. The name is used to reference the trust anchor that is defined.

   Enter the key store password, path, and key store type.

   Click the trust anchor name link to edit the selected trust anchor.

   Click **Remove** to remove the selected trust anchor or anchors.

   When you start the application, the configuration is validated in the run time while the binding information is loading.
8. Save the changes.

Results

This procedure defines trust anchors that can be used by the request receiver or the response receiver (if the web services is acting as client) to verify the signer certificate.

Example

The request receiver or the response receiver (if the web service is acting as a client) uses the defined trust anchor to verify the signer certificate. The trust anchor is referenced using the trust anchor name.

What to do next

To complete the signing information configuration process for request receiver, complete the following tasks:

1. Configuring the server for request digital signature verification: Verifying the message parts
2. Configuring the server for request digital signature verification: choosing the verification method

To complete the process for the response receiver, if the web services is acting as client, complete the following tasks:

1. Configuring the client for response digital signature verification: verifying the message parts
2. Configuring the client for response digital signature verification: choosing the verification method

Configuring the client-side collection certificate store using the administrative console

You can configure the client-side collection certificate store by using the administrative console.

About this task

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6 and later applications.

A *collection certificate store* is a collection of non-root, certificate authority (CA) certificates and certificate revocation lists (CRLs). This collection of CA certificates and CRLs are used to check the signature of a digitally signed SOAP message.

You can configure the collection certificate either by using the assembly tools or the WebSphere Application Server administrative console. Complete the following steps to configure the client-side collection certificate store using the administrative console.

Procedure

1. Connect to the WebSphere Application Server administrative console.

   You can connect to the administrative console by typing `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.
2. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
3. Under Manage modules, click *URI_name*.
4. Under Web Services Security Properties, click **Web services: Client security bindings** to add the collection certificate store to the client security bindings. If you do not see any entries, return to the assembly tool and configure the security extensions for either the client or the server.

   To configure the security extensions for the client, see the following topics:
   - Configuring the client for response digital signature verification: verifying the message parts
   - Configuring the client for response digital signature verification: choosing the verification method
5. Under Response receiver binding, click **Edit** to edit the client security bindings.
6. Click **Collection certificate store**.
7. Click a Certificate store name to edit an existing certificate store or click **New** to add a new certificate store name.
8. Enter a name in the Certificate store name field. The name entered in this field is a name that is referenced in the Certificate store field on the Signing information configuration page.
9. Leave the Certificate store provider field value as `IBMCertPath`.
10. Click **Apply**.
11. Under Additional properties, click **X.509 certificates** > **New**.
12. Enter the path to your certificate store. For example, the path might be: `${USER_INSTALL_ROOT}/etc/ws-security/samples/intca2.cer`. If you have any additional certificate store paths to enter, click **New** and add the path names.
13. Click **OK**.

Configuring the server-side collection certificate store using the administrative console

You can configure the collection certificate either by using an assembly tool or the WebSphere Application Server administrative console.

About this task

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6 and later applications.

A *collection certificate store* is a collection of non-root, certificate authority (CA) certificates and certificate revocation lists (CRLs). This collection of CA certificates and CRLs is used to check the signature of a digitally signed SOAP message.

Complete the following steps to configure the server-side collection certificate store using the administrative console.

Procedure

1. Connect to the WebSphere Application Server administrative console.

   You can connect to the administrative console by typing `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.
2. Click **Applications > Application Types > WebSphere enterprise applications > *application_name***.
3. Under Manage modules, click ***URI_name***
4. Under Web Services Security Properties, click **Web services: server security bindings** to add the collection certificate store to the server security bindings. If you do not see any entries, return to the assembly tool and configure the security extensions for the server.

   To configure the security extensions for the server, see the following topics:
   - Configuring the server for request digital signature verification: Verifying the message parts
   - Configuring the server for request digital signature verification: choosing the verification method
5. Click **Edit** under Request Receiver Binding to edit the server security bindings.
6. Click **Collection certificate store**.
7. Click a Certificate store name to edit an existing certificate store or click **New** to add a new certificate store name.
8. Enter a name in the Certificate store name field. The name entered in this field is a name that is referenced in the Certificate store field on the Signing information configuration page.
9. Leave the Certificate store provider field as `IBMCertPath`.
10. Click **Apply**.
11. Under Additional Properties, click **X.509 Certificates > New**.
12. Enter the path to your certificate store. For example, the path might be: `${USER_INSTALL_ROOT}/etc/ws-security/samples/intca2.cer`. If you have any additional certificate store paths to enter, click **New** and add the path names.
13. Click **OK**.

Configuring default collection certificate stores at the server level in the WebSphere Application Server administrative console

You can define a single collection certificate store for all of the applications that need to use the same certificates. Use the WebSphere Application Server administrative console to configure the default collection certificate store at the server level.

About this task

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

A *collection certificate store* is a collection of non-root, certificate authority (CA) certificates and certificate revocation lists (CRLs). This collection of CA certificates and CRLs are used to check the signature of a digitally signed SOAP message. A certificate store typically refers to a certificate store located in the file system. The location of the certificate store can vary from machine to machine, so you might configure a default collection certificate store for a specific machine and reference it from within the signing information. The signing information is found within the binding configurations of any application installed on the machine. This suggestion enables you to define a single collection certificate store for all of the applications that need to use the same certificates. You also can specify the default binding information at the cell level.

Complete the following steps to configure the default collection certificate store at the server level using the WebSphere Application Server administrative console:

Procedure

1. Connect to the administrative console.

   You can access the administrative console by typing `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.
2. Click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name***.
3. Under Security, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using Websphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.
4. Under Additional properties, click **Collection certificate store**.
5. Enter a name in the **Certificate store name** field. This name is referenced in the **Certificate store** field on the Signing information configuration page.
6. Leave the **Certificate store provider** field value as `IBMCertPath`.
7. Click **Apply**.
8. Under Additional properties, click **X.509 certificates** > **New**.
9. Enter the path to your certificate store. For example, the path might be: `${USER_INSTALL_ROOT}/etc/ws-security/samples/intca2.cer`.

   If you have any additional certificate store paths to enter, click **New** and add the path names.
10. Click **OK**.

Configuring key locators using the administrative console

You can configure binding information and key locators using the WebSphere Application Server administrative console.

About this task

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

This task provides instructions on how to configure key locators using the WebSphere Application Server administrative console. You can configure binding information in the administrative console. You must use an assembly tool to configure extensions. The following steps are used to configure a key locator in the administrative console for a specific application:

Procedure

1. Open the administrative console.

   Type `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.
2. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
3. Under Related Items, click either **Web Modules** or **EJB Modules**, depending on the type of module you are securing.
4. Click the name of the module you are securing.
5. Under Additional Properties, click either **Web services: Client security bindings** or **Web services: Server security bindings**, depending on whether you are adding the key locator to the client security bindings or to the server security bindings. If you do not see any entries, return to the assembly tool and configure the security extensions.

6. Edit the Request Sender Binding, Response Receiver Binding, Request Receiver Binding, or Response Sender Binding.
   - If you are editing your client security bindings, click **Edit** for either the Request Sender Binding or the Response Receiver Binding.
   - If you are editing your server security bindings, click **Edit** for either the Request Receiver Binding or the Response Sender Binding.
7. Click **Key Locators**.
8. Click **New** to configure a new key locator, select the box next to a key locator name and click **Delete** to delete a key locator, or click the name of a key locator to edit its configuration. If you are configuring a new key locator or editing an existing one, complete the following steps:
   a. Specify a name for the key locator in the **Key Locator Name** field.
   b. Specify a name for the key locator class implementation in the **Key Locator Classname** field. WebSphere Application Server has the following default key locator class implementations:

      com.ibm.wsspi.wssecurity.config.WSIdKeyStoreMapKeyLocator
      This class is used by the response sender to map an authenticated identity to a key. If encryption is used, this class is used to locate a key to encrypt the response message. The `com.ibm.wsspi.wssecurity.config.WSIdKeyStoreMapKeyLocator` class has the capability to map an authenticated identity from the invocation credential of the current thread to a key that is used to encrypt the message. If an authenticated identity is present on the current thread, the class maps the ID to the mapped name. For example, `user1` is mapped to `mappedName_1`. Otherwise, `name="default"`. When a matching key is not found, the authenticated identity is mapped to the default key specified in the binding file.

      com.ibm.wsspi.wssecurity.config.KeyStoreKeyLocator
      This class is used by the response receiver, the request sender, and the request receiver to map a name to an alias. Encryption uses this class to obtain a key to encrypt a message and digital signature uses this class to obtain a key to sign a message. The `com.ibm.wsspi.wssecurity.config.KeyStoreKeyLocator` class maps a logical name to a key alias in the key store file. For example, key #105115176771 maps to `CN=Alice, O=IBM, C=US`.

   c. Specify the password used to access the key store password in the **Key Store Password** field. This field is optional because the key locator does not use a key store.
   d. Specify the path name used to access the key store in the **Key Store Path** field. This field is optional because the key locator does not use a key store. Use `${USER_INSTALL_ROOT}` because this path expands to the WebSphere Application Server path on your machine.
   e. Select a keystore type from the **Key Store Type** field. This field is optional because the key locator does not use a key store. Use the `JKS` option if you are not using the Java Cryptography Extensions (JCE) policy and use `JCEKS` if you are using the JCE policy.

Configuring the security bindings on a server acting as a client using the administrative console

Use the web services client editor within an assembly tool to include the binding information, that describes how to run the security specifications found in the extensions, in the client enterprise archive (EAR) file.

About this task

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

When configuring a client for Web Services Security, the bindings describe how to run the security specifications found in the extensions. Use the web services client editor within an assembly tool to include the binding information in the client enterprise archive (EAR) file.

You can configure the client-side bindings from a pure client accessing a web service or from a web service accessing a downstream web service. Complete the following steps to find the location in which to edit the client bindings from a web service that is running on the server. When a web service communicates with another web service, you must configure client bindings to access the downstream web service.

Procedure

1. Deploy the web service using the WebSphere Application Server administrative console. Click **Applications** > **Install New Application**.

   You can access the administrative console by typing `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.

   For more information, read about installing a new application.
2. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
3. Under Manage modules, click ***URI_name***.
4. Under Web Services Security Properties, click **Web Services: Client security bindings**. A table displays with the following columns:
   - Component Name
   - Port
   - Web Service
   - Request Sender Binding
   - Request Receiver Binding
   - HTTP Basic Authentication
   - HTTP SSL Configuration

   For Web Services Security, you must edit the request sender binding and response receiver binding configurations. You can use the defaults for some of the information at the server level. Default bindings are convenient because you can configure commonly reused elements such as key locators once and then reference their aliases in the application bindings.
5. View the default bindings for the server using the administrative console by clicking **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** . Under Additional Properties, click **JAX-WS and JAX-RPC security runtime**.

   Note: In a mixed node cell with a server using WebSphere Application Server version 6.1 or earlier, click **Web services: Default bindings for Web Services Security**.

   You can configure the following sections. These topics are discussed in more detail in other sections of the documentation.
   - Request sender binding
     - Signing parameter configuration settings
     - “Encryption information configuration settings: Methods” on page 3363
     - “Key locator configuration settings” on page 3408
     - “Login bindings configuration settings” on page 3458
   - Response receiver binding
     - “Signing information configuration settings” on page 3299
     - “Encryption information configuration settings: Message parts” on page 3358
     - “Trust anchor configuration settings” on page 3419

– “Collection certificate store configuration settings” on page 3426
– “Key locator configuration settings” on page 3408

What to do next

Important: When configuring the security request sender binding configuration, you must synchronize the information used to perform the specified security with the security request receiver binding configuration, which is configured in the server EAR file. These two configurations must be synchronized in all respects because there is no negotiation during run time to determine the requirements of the server. For example, when configuring the encryption information in the security request sender binding configuration, you must use the public key from the server for encryption. Therefore, the key locator that you choose must contain the public key from the server configuration. The server must contain the private key to decrypt the message. This example illustrates the important relationship between the client and server configuration. Additionally, when configuring the security response receiver binding configuration, the server must send the response using security information known by this client security response receiver binding configuration.

The following table shows the related configurations between the client and the server. The client request sender and the server request receiver are relative configurations that must be synchronized with each other. The server response sender and the client response receiver are related configurations that must be synchronized with each other. Note that related configurations are end points for any request or response. One end point must communicate its actions with the other end point because run time requirements are not required.

Table 321. Related configurations. The configurations must be synchronized with each other.

Client configuration	Server configuration
Request sender	Request receiver
Response receiver	Response sender

Configuring the server security bindings using the administrative console

Use the WebSphere Application Server administrative console to edit bindings for a web service after these bindings are deployed on a server.

About this task

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

Create an Enterprise JavaBeans (EJB) file Java archive (JAR) file or web application archive (WAR) file containing the security binding file (`ibm-webservices-bnd.xmi`) and the security extension file (`ibm-webservices-ext.xmi`). If this archive is acting as a client to a downstream service, you also need the client-side binding file (`ibm-webservicesclient-bnd.xmi`) and the client-side extension file (`ibm-webservicesclient-ext.xmi`). These files are generated using the WSDL2Java command. For more information, read about the WSDL2Java command for JAX-RPC applications. You can edit these files using the Web Services Editor in the assembly tools. For more information, read about assembly tools.

When configuring server-side security for Web Services Security, the security extensions configuration specifies what security is to be performed while the security bindings configuration indicates how to perform what is specified in the security extensions configuration. You can use the defaults for some elements at the cell and server levels in the bindings configuration, including key locators, trust anchors, the collection certificate store, trusted ID evaluators, and login mappings and reference them from the WAR and JAR binding configurations.

The following steps describe how to edit bindings for a web service after these bindings are deployed on a server. When one web service communicates with another web service, you also must configure the client bindings to access the downstream web service.

Procedure

1. Deploy the web service using the WebSphere Application Server administrative console.

   Type `http://server_name:port_number/ibm/console` in your web browser unless you have changed the port number.

   After you log into the administration console, click **Applications** > **Install new application** to deploy the web service. For more information, read about installing enterprise application files with the console.
2. After you deploy the web service, click **Applications** > **Enterprise applications** > ***application_name*** .
3. Under Manage modules, click ***URI_name***.
4. Under Web Services Security Properties, click **Web services: client security bindings** for outbound requests and inbound responses. Click **Web services: server security bindings** for inbound requests and outbound responses.
5. If you click **Web services: server security bindings**, the following sections can be configured. These topics are discussed in more detail in other sections of the documentation.
   - Request receiver binding
     - Signing information
     - Encryption information
     - Trust anchors
     - Collection certificate store
     - Key locator
     - Trusted ID evaluator
     - Login mappings
   - Response sender binding
     - Signing parameters
     - Encryption information
     - Key locator

Configuring XML encryption for Version 5.x web services with the administrative console

XML encryption is one method that WebSphere® Application Server provides to secure web services. You can use XML encryption in conjunction with XML digital signature to scramble the content while verifying the authenticity of the message sender. Using XML encryption, you can encrypt an XML element, the content of an XML element, or arbitrary data such as an XML document.

Login bindings configuration settings

Use this page to specify the Java Authentication and Authorization Service (JAAS) login configuration settings that are used to validate security tokens within incoming messages.

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications. Version 5.*x* applications are based on Java 2 platform, Enterprise Edition (J2EE) 1.3.

The pluggable token uses the Java Authentication and Authorization Service (JAAS) CallBackHandler (javax.security.auth.callback.CallBackHandler) interface to generate the token that is inserted into the message. The following list describes the CallBack support implementations:

com.ibm.wsspi.wssecurity.auth.callback.BinaryTokenCallback
This implementation is used for generating binary tokens inserted as `<wsse:BinarySecurityToken/@ValueType>` in the message.

javax.security.auth.callback.NameCallback and javax.security.auth.callback.PasswordCallback
This implementation is used for generating user name tokens inserted as `<wsse:UsernameToken>` in the message.

com.ibm.wsspi.wssecurity.auth.callback.XMLTokenSenderCallback
This implementation is used to generate Extensible Markup Language (XML) tokens and is inserted as the `<SAML: Assertion>` element in the message.

com.ibm.wsspi.wssecurity.auth.callback.PropertyCallback
This implementation is used to obtain properties that are specified in the binding file.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_file_name***. Under Web Services Security Properties, click **Web Services: Client security bindings**.
3. Under Request Sender Bindings, click **Edit**.
4. Under Additional properties, click **Login binding**.

If the encryption information is not available, select **None**.

If the encryption information is available, select **Dedicated login binding** and specify the configuration in the following fields:

Authentication method:

Specifies the unique name for the authentication method.

You can uses any string to name the authentication method. However, the string must match the element in the server-level configuration. The following words are reserved by WebSphere Application Server:

BasicAuth
This method uses both a user name and a password.

IDAssertion
This method uses a user name, but it requires that additional trust is established by the receiving server using a trusted ID evaluator mechanism.

Signature
This method uses the distinguished name (DN) of the signer.

LTPA This method validates the token.

Callback handler:

Specifies the name of the callback handler. The callback handler must implement the `javax.security.auth.callback.CallbackHandler` interface.

Basic authentication user ID:

Specifies the user name for basic authentication. With the basic authentication method, you can define a user name and a password in the binding file.

Basic authentication password:

Specifies the password for basic authentication.

Token type URI:

Specifies the namespace Uniform Resource Identifiers (URI), which denotes the type of security token that is accepted.

The value of this field if is impacted by the following conditions:

- If binary security tokens are accepted, the value denotes the ValueType attribute in the element. The ValueType element identifies the type of security token and its namespace.
- If Extensible Markup Language (XML) tokens are accepted, the value denotes the top-level element name of the XML token.
- The Token type URI field is ignored if the reserved words, which are listed in the description of the Authentication method field, are specified.

This information is inserted as `<wsse:BinarySecurityToken>/ValueType` for the <SAML: Assertion> XML token.

Token type local name:

Specifies the local name of the security token type. For example, X509v3.

The value of this field if is impacted by the following conditions:

- If binary security tokens are accepted, the value denotes the ValueType attribute in the element. The ValueType element identifies the type of security token and its namespace.
- If Extensible Markup Language (XML) tokens are accepted, the value denotes the top-level element name of the XML token.
- The Token type URI field is ignored if the reserved words, which are listed in the description of the Authentication method field, are specified.

This information is inserted as `<wsse:BinarySecurityToken>/ValueType` for the <SAML: Assertion> XML token.

Request sender binding collection

Use this page to specify the binding configuration to send request messages for Web Services Security.

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications. Version 5.*x* applications are based on Java 2 platform, Enterprise Edition (J2EE) 1.3.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_file_name***.
3. Under Web Services Security Properties, click **Web services: Client security bindings**.
4. Under Request sender binding, click **Edit**.

Web Services Security namespace: Specifies the namespace that is used by Web Services Security to send a request. However, this field configures the namespace value only and does not enforce the semantics of the specification related to the namespace. Web Services Security uses the processing semantic only in draft 13 of the OASIS specification. The following schemas are available:

- http://schemas.xmlsoap.org/ws/2003/06/secext
- http://schemas.xmlsoap.org/ws/2002/07/secext
- http://schemas.xmlsoap.org/ws/2002/04/secext
- None

The namespace used by the response sender is based on the namespace of the incoming message in the request receiver.

Signing information:

Specifies the configuration for the signing parameters. Signing information is used to sign and validate parts of the message including the body and time stamp.

You can also use these parameters for X.509 validation when the Authentication method is `IDAssertion` and the ID Type is `X509Certificate`, in the server-level configuration. In such cases, you must fill in the Certificate Path fields only.

Encryption information:

Specifies the configuration for the encrypting and decrypting parameters. Encryption information is used for encrypting and decrypting various parts of a message, including the body and user name token.

Key locators:

Specifies a list of key locator objects that retrieve the keys for digital signature and encryption from a keystore file or a repository. The key locator maps a name or a logical name to an alias or maps an authenticated identity to a key. This logical name is used to locate a key in a key locator implementation.

Login mappings:

Specifies a list of configurations for validating tokens within incoming messages.

Login mappings map the authentication method to the Java Authentication and Authorization Service (JAAS) configuration.

To configure JAAS, complete the following steps:

1. Click **Security** > **Global security**.
2. Under the Java Authentication and Authorization Service field, select **Application logins** or **System logins**.

Request receiver binding collection

Use this page to specify the binding configuration to receive request messages for Web Services Security.

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications. Version 5.*x* applications are based on Java 2 platform, Enterprise Edition (J2EE) 1.3.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name***.
2. Under Modules, click **Manage modules** > ***URI_file_name***.
3. Under Web Services Security Properties, click **Web services: Server security bindings**.
4. Under Request receiver binding, click **Edit**.

Signing information:

Specifies the configuration for the signing parameters. Signing information is used to sign and validate parts of a message including the body, the timestamp, and the user name token.

You also can use these parameters for X.509 certificate validation when the authentication method is `IDAssertion` and the ID Type is `X509Certificate` in the server-level configuration. In such cases, you must fill in the Certificate Path fields only.

Encryption information:

Specifies the configuration for the encrypting and decrypting parameters. This configuration is used to encrypt and decrypt parts of the message that include the body and the user name token.

Trust anchors:

Specifies a list of keystore objects that contain the trusted root certificates that are issued by a certificate authority (CA).

The certificate authority authenticates a user and issues a certificate. The CertPath API uses the certificate to validate the certificate chain of incoming, X.509-formatted security tokens or trusted, self-signed certificates.

Collection certificate store:

Specifies a list of the untrusted, intermediate certificate files.

The collection certificate store contains a chain of untrusted, intermediate certificates. The CertPath API attempts to validate these certificates, which are based on the trust anchor.

Key locators:

Specifies a list of key locator objects that retrieve the keys for digital signature and encryption from a keystore file or a repository. The key locator maps a name or a logical name to an alias or maps an authenticated identity to a key. This logical name is used to locate a key in a key locator implementation.

Trusted ID evaluators:

Specifies a list of trusted ID evaluators that determine whether to trust the identity-asserting authority or message sender.

The trusted ID evaluators are used to authenticate additional identities from one server to another server. For example, a client sends the identity of user A to server 1 for authentication. Server 1 calls downstream to server 2, asserts the identity of user A, and includes the user name and password of server 1. Server 2 attempts to establish trust with server 1 by authenticating its user name and password and checking the trust based on the TrustedIDEvaluator implementation. If the authentication process and the trust check are successful, server 2 trusts that server 1 authenticated user A and a credential is created for user A on server 2 to invoke the request.

Login mappings:

Specifies a list of configurations for validating tokens within incoming messages.

Login mappings map the authentication method to the Java Authentication and Authorization Service (JAAS) configuration.

To configure JAAS, complete the following steps:

1. Click **Security** > **Global security**.
2. Under the Java Authentication and Authorization Service, click **Application logins** or **System logins**.

Response sender binding collection

Use this page to specify the binding configuration for sender response messages for Web Services Security.

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications. Version 5.*x* applications are based on Java 2 platform, Enterprise Edition (J2EE) 1.3.

To view this administrative console page, complete the following steps:

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***application_name*** .
2. Under Modules, click **Manage modules** > ***URI_file_name***.
3. Under Web Services Security Properties, click **Web services: Server security bindings**.
4. Under Response sender binding, click **Edit**.

Signing information:

Specifies the configuration for the signing parameters.

You also can use these parameters for X.509 certificate validation when the authentication method is IDAssertion and the ID Type is X509Certificate in the server-level configuration. In such cases, you must fill-in the Certificate Path fields only.

Encryption information:

Specifies the configuration for the encryption and decryption parameters.

Key locators:

Specifies a list of key locator objects that retrieve the keys for a digital signature and encryption from a keystore file or a repository. The key locator maps a name or logical name to an alias or maps an authenticated identity to a key. This logical name is used to locate a key in a key locator implementation.

Response receiver binding collection

Use this page to specify the binding configuration for receiver response messages for Web Services Security.

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications. Version 5.*x* applications are based on Java 2 platform, Enterprise Edition (J2EE) 1.3.

To view this administrative console page, complete the following steps:

1. Click **Applications > Application Types > WebSphere enterprise applications***application_name*.
2. Under Modules, click **Manage modules > *URI_file_name* > Web Services: Client security bindings**.
3. Under Response receiver binding, click **Edit**.

Signing information:

Specifies the configuration for the signing parameters. Signing information is used to sign and to validate parts of the message including the body and the timestamp.

You can also use these parameters for X.509 validation when the authentication method is IDAssertion and the ID type is X509Certificate, in the server-level configuration. In such cases, you must fill in the certificate path fields only.

Encryption information:

Specifies the configuration for the encryption and decryption parameters.

Encryption information is used for encrypting and decrypting various parts of a message, including the body and the user name token.

Trust anchors:

Specifies a list of keystore objects that contain the trusted root certificates that are self-signed or issued by a certificate authority.

The certificate authority authenticates a user and issues a certificate. After the certificate is issued, the keystore objects, which contain these certificates, use the certificate for certificate path or certificate chain validation of incoming X.509-formatted security tokens.

Collection certificate store:

Specifies a list of the untrusted, intermediate certificate files.

The collection certificate store contains a chain of untrusted, intermediate certificates. The CertPath API attempts to validate these certificates, which are based on the trust anchor.

Key locators:

Specifies a list of key locator objects that retrieve the keys for a digital signature and encryption from a keystore file or a repository.

The key locator maps a name or a logical name to an alias or maps an authenticated identity to a key. This logical name is used to locate a key in a key locator implementation.

Configuring pluggable tokens using the administrative console

You can configure the client-side request sender (`ibm-webservicesclient-bnd.xmi` file) or server-side request receiver (`ibm-webservices-bnd.xmi` file) by using the WebSphere Application Server administrative console.

Before you begin

Important: There is an important distinction between Version 5.*x* and Version 6 and later applications. The information in this article supports Version 5.*x* applications only that are used with WebSphere Application Server Version 6.0.*x* and later. The information does not apply to Version 6.0.*x* and later applications.

Prior to completing these steps, it is assumed that you have already created a web service that is based on the Java Platform, Enterprise Edition (Java EE) specification. See either of the following topics for an introduction of how to manage Web Services Security binding information for the server:

- Configuring the server security bindings using an assembly tool
- "Configuring the server security bindings using the administrative console" on page 3457

About this task

This document describes how to configure a pluggable token in the request sender (`ibm-webservicesclient-ext.xmi` and `ibm-webservicesclient-bnd.xmi` file) and request receiver (`ibm-webservices-ext.xmi` and `ibm-webservices-bnd.xmi` file).

Important: The pluggable token is required for the request sender and request receiver as they are a pair. The request sender and the request receiver must match for a request to be accepted by the receiver.

Prior to completing these steps, it is assumed that you deployed a web services-enabled enterprise application to the WebSphere Application Server.

Use the following steps to configure the client-side request sender (`ibm-webservicesclient-bnd.xmi` file) or server-side request receiver (`ibm-webservices-bnd.xmi` file) using the WebSphere Application Server administrative console.

1. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***enterprise_application***.
2. Under Modules, click **Manage modules** > ***URI_name***. The *URI* is the web services-enabled module.
   a. Under Web Services Security Properties, click **Web services: client security bindings** to edit the response sender binding information, if web services are acting as client.
      1) Under Response sender binding, click **Edit**.
      2) Under Additional Properties, click **Login binding**.
      3) Select **Dedicated login binding** to define a new login binding.
         a) Enter the authentication method, this must match the authentication method defined in IBM extension deployment descriptor. The authentication method must be unique in the binding file.
         b) Enter an implementation of the JAAS `javax.security.auth.callback.CallbackHandler` interface.
         c) Enter the basic authentication information (User ID and Password) and the basic authentication information is passed to the construct of the CallbackHandler implementation. The usage of the basic authentication information is up to the implementation of the CallbackHandler.
         d) Enter the token value type, it is optional for BasicAuth, Signature and IDAssertion authentication methods but required for any other authentication method. The token value type is inserted into the <wsse:BinarySecurityToken>@ValueType for binary security token and used as the namespace of the XML based token.
         e) Click **Properties**. Define the property with name and value pairs. These pairs are passed to the construct of the CallbackHandler implementation as `java.util.Map`.

      Select **None** to deselect the login binding.
   b. Under Web Services Security Properties, click **Web services: server security bindings** to edit the request receiver binding information.
      1) Under Request Receiver Binding, click **Edit**.
      2) Under Additional Properties, click **Login mappings**.
      3) Click **New** to create new login mapping.

a) Enter the authentication method, this must match the authentication method defined in the IBM extension deployment descriptor. The authentication method must be unique in the login mapping collection of the binding file.

b) Enter a JAAS Login Configuration name. The JAAS Login Configuration must be defined under **Security** > **Global security**. Under Authentication, click **Java Authentication and Authorization Service** > **Application logins**. For more information, read about configuring programmatic logins for Java Authentication and Authorization Service.

c) Enter an implementation of the `com.ibm.wsspi.wssecurity.auth.callback.CallbackHandlerFactory` interface. This is a mandatory field.

d) Enter the token value type, it is optional for BasicAuth, Signature and IDAssertion authentication methods but required for any other authentication method. The token value type is used to validate against the <wsse:BinarySecurityToken>@ValueType for binary security token and against the namespace of the XML based token.

e) Enter the name and value pairs for the "Login Mapping Property" by clicking **Properties** . These name and value pairs are available to the JAAS Login Module or Modules by `com.ibm.wsspi.wssecurity.auth.callback.PropertyCallback` JAAS Callback. **Note**: This is true when editing existing login mappings but not when creating new login mappings.

f) Enter the name and value pairs for the "Callback Handler Factory Property", these name and value pairs is passed as `java.util.Map` to the `com.ibm.wsspi.wssecurity.auth.callback.CallbackHandlerFactory.init()` method. The usage of these name and value pairs is up to the `CallbackHandlerFactory` implementation.

c. Click authentication method link to edit the selected login mapping.

d. Click **Remove** to remove the selected login mapping or mappings.

3. Click **Save**.

Results

The previous steps define how to configure the request sender to create security tokens in the SOAP message and the request receiver to validate the security tokens found in the incoming SOAP message. WebSphere Application Server supports pluggable security tokens.

You can use the authentication method defined in the login bindings and login mappings to generate security tokens in the request sender and validate security tokens in the request receiver.

What to do next

After you have configured pluggable tokens, you must configure both the client and the server to support pluggable tokens. See the following topics to configure the client and the server:

- Configuring the client for LTPA token authentication: specifying LTPA token authentication
- Configuring the client for LTPA token authentication: collecting the authentication method information
- Configuring the server to handle LTPA token authentication information
- Configuring the server to validate LTPA token authentication information

Chapter 37. Administering web services - Transaction support (WS-Transaction)

WS-Transaction is an interoperability standard that includes the WS-AtomicTransaction, WS-BusinessActivity, and WS-Coordination specifications. The Web Services Atomic Transaction (WS-AT) support in the application server provides transactional quality of service to the web services environment. Distributed web services applications, and the resources they use, can take part in distributed global transactions. With Web Services Business Activity (WS-BA) support in the application server, web services on different systems can coordinate activities that are more loosely coupled than atomic transactions. Such activities can be difficult or impossible to roll back atomically, and therefore require a compensation process if an error occurs. Web Services Coordination (WS-COOR) specifies a CoordinationContext and a Registration service with which participant web services can enlist to take part in the protocols that are offered by specific coordination types.

Using WS-Transaction policy to coordinate transactions or business activities for web services

You can use WS-Transaction policy to configure how a Java API for XML Web Services (JAX-WS) service or client handles Web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) context.

About this task

The Web Services Atomic Transaction (WS-AT) support in the application server provides transactional quality of service to the web services environment. Distributed web services applications, and the resources they use, can take part in distributed global transactions. With Web Services Business Activity (WS-BA) support in the application server, web services on different systems can coordinate activities that are more loosely coupled than atomic transactions. Such activities can be difficult or impossible to roll back atomically, and therefore require a compensation process if an error occurs.

Procedure

- Configure a JAX-WS client for WS-Transaction context.
- Configure a JAX-WS web service for WS-Transaction context.
- Configure the WS-Transaction policy.
- Configure a WS-Transaction policy set by using wsadmin scripting.
- Configure transaction properties for an application server.
- Configure the WS-Transaction specification level by using wsadmin scripting.
- Configure WS-Transaction support in a secure environment.
- Configure an intermediary node for web services transactions.
- Enable WebSphere Application Server to use an intermediary node for web services transactions.
- Configure a server to use business activity support.

Configuring a JAX-WS client for WS-Transaction context

You can configure the way that a Java API for XML Web Services (JAX-WS) client handles Web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) context by configuring the Web Services Transaction (WS-Transaction) policy type. You can specify that the client must send context, can send context if it is available, or must not send context.

Before you begin

A JAX-WS client must be installed.

About this task

You can configure a WS-Transaction policy set by using the administrative console, as described in this task, or you can configure a WS-Transaction policy set by using wsadmin scripting.

To configure a JAX-WS client for WS-Transaction context by using the administrative console, complete the following steps.

Procedure

1. Create a new policy set, or copy and rename an existing policy set. You can copy an existing user-defined policy set, or one of the WS-Transaction default policy sets (WSTransaction or SSL WSTransaction). See “Creating policy sets using the administrative console” on page 2668.
2. Check that your policy set includes the WS-Transaction policy type. If necessary, add the WS-Transaction policy type. See “Adding policies to policy sets using the administrative console” on page 2712.
3. Configure the WS-Transaction policy.
4. Associate the policy set with the JAX-WS client. See Managing policy sets and bindings for service clients.
5. Choose a WS-Policy application rule that includes the configured policy of the client, that is, client only or client and provider. See “Configuring the client policy to use a service provider policy” on page 3107.
6. Save your changes to the master configuration.

Results

The JAX-WS client is configured to use WS-Transaction context in the way that you specified.

Configuring a JAX-WS web service for WS-Transaction context

You can configure the way that a Java API for XML Web Services (JAX-WS) service handles Web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) context by configuring the Web Services Transaction (WS-Transaction) policy type. You can specify that the web service must receive context, can receive context if it is available, or must not receive context.

Before you begin

A JAX-WS client must be installed.

About this task

You can configure a WS-Transaction policy set by using the administrative console as described in this task, or you can configure a WS-Transaction policy set by using wsadmin scripting.

To configure a JAX-WS service for WS-Transaction context by using the administrative console, complete the following steps.

Procedure

1. Create a new policy set, or copy and rename an existing policy set. You can copy an existing user-defined policy set, or one of the WS-Transaction default policy sets (WSTransaction or SSL WSTransaction). See “Creating policy sets using the administrative console” on page 2668.
2. Check that your policy set includes the WS-Transaction policy type. If necessary, add the WS-Transaction policy type. See “Adding policies to policy sets using the administrative console” on page 2712.
3. Configure the WS-Transaction policy.

4. Associate the policy set with the JAX-WS web service, endpoint, or operation for which the specified behavior is required. See Managing policy sets and bindings for service providers.
5. Save your changes to the master configuration.

Results

The JAX-WS web service, endpoint, or operation is configured to use WS-Transaction context in the way that you specified.

Configuring a WS-Transaction policy set by using wsadmin scripting

You can configure the way that a Java API for XML Web Services (JAX-WS) client or web service handles Web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) context by configuring the Web Services Transaction (WS-Transaction) policy type. You can specify that the client or service must use context, can use context if it is available, or must not use context. Use command scripts to configure a policy set for web services transactions.

About this task

You can configure a WS-Transaction policy set by using wsadmin scripting as described in this task, or you can configure a WS-Transaction policy set by using the administrative console.

Procedure

1. Start the wsadmin scripting client if it is not already running.
2. Use the createPolicySet command to create a new policy set, or the copyPolicySet command to copy and rename an existing policy set. You can copy an existing user-defined policy set, or one of the WS-Transaction default policy sets (WSTransaction or SSL WSTransaction).
3. Check that your policy set includes the WS-Transaction policy type. If necessary, add the WS-Transaction policy type. For example:

```
AdminTask.importPolicySet('[-defaultPolicySet WSTransaction]')
AdminTask.addPolicyType('[-policySet policy_set_name
-policyType WSTransaction -enabled true]')
```

4. Use the setPolicyType command to configure the WS-Transaction policy type attributes. The WS-Transaction policy type has the following attributes:
   - ATAssertion
   - BAAtomicOutcomeAssertion

   Each attribute can have the value supports, mandatory, or never. For detailed information about these configurable attributes, see the topic about WS-Transaction policy settings. For example:

```
AdminTask.setPolicyType('[-policySet policy_set_name
-policyType WSTransaction
-attributes "[ [BAAtomicOutcomeAssertion mandatory] [ATAssertion supports] ]"]
-replace')
```

5. Save your changes to the master configuration. For example, enter the following command:

```
AdminConfig.save()
```

What to do next

You are now ready to associate the policy set with the JAX-WS client, or with the JAX-WS web service, endpoint, or operation.

Configuring Web Services Transaction support in a secure environment

If you use Web Services Atomic Transaction (WS-AT) or Web Services Business Activity (WS-BA) support when administrative security is enabled, you might have to change the default transaction service configuration. You can disable the transaction coordination authorization setting, create a new web container transport chain, or do both.

About this task

You might disable transaction coordination authorization if you want to interoperate with other servers and you do not want to set up security for the transaction manager to support the Common Criteria EAL4 evaluated configuration. When transaction coordination authorization is disabled, WebSphere Application Server does not automatically reject secure WS-Transactions protocol messages.

You might configure a new web container transport chain for use by WS-Transactions in the following situations:

- You want to use an alternative port number for WS-AT or WS-BA protocol messages.
- You want to interoperate with a non-WebSphere Application Server that requires client certificate authentication on the Secure Sockets Layer (SSL) connection that is used for protocol messages.

The transaction service, by default, selects a suitable web container transport chain from the list of those configured and uses it for protocol messages. You can configure a new transport chain and specify your own settings. For example, you can specify an alternative SSL configuration that requires client certificate authentication, which is then used specifically for WS-Transactions protocol messages.

Procedure

1. Optionally, use the following steps to disable transaction coordination authorization.
   a. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Transaction Service**.
   b. Clear the **Enable transaction coordination authorization** check box.
   c. Click **Apply** or **OK**.
   d. Save your changes to the master configuration.
2. Optionally, use the following steps to create a new web container transport chain.
   a. In the administrative console, click **Servers** > **Application servers** > ***server_name*** > **[Container Settings] Web Container Settings** > **Web container transport chains**.
   b. Click **New** to create a new transport chain.
   c. Type a name for the transport chain.
   d. From the Transport chain template list, select an appropriate template.
   e. Click **Next** to select a new port for the chain.
   f. Type a name, host, and port number for the port. For a secure chain, the host must match the common name in the certificate that is used.
   g. Click **Next**, confirm the settings, then click **Finish**.
   h. Save your changes to the master configuration.
   i. If necessary, create a new SSL configuration and associate it with the SSL channel associated with your new chain. For more information, see “Creating a Secure Sockets Layer configuration” on page 1741. You are now ready to configure the transaction service to use the new transport chain.
   j. Click **Servers** > **Application servers** > ***server_name*** > **[Container Settings] Container Services** > **Transaction Service**.
   k. In the External WS-Transaction HTTP(S) URL prefix section, click **Select prefix**, then select the web container transport chain that you have just created from the list.

If you are using an intermediary, such as an HTTP proxy, in front of the application server, click **Specify custom prefix**, then type the external endpoint URL information for the intermediary node in the field. For more information, see “Enabling WebSphere Application Server to use an intermediary node for web services transactions” on page 3472.

l. Click **Apply** or **OK**, then save your changes to the master configuration.

3. After you save all the configuration changes, restart the server for the changes to take effect.

Results

You configured your system to use WS-AT or WS-BA in a secure environment.

Configuring an intermediary node for web services transactions

Intermediary nodes allow the exchange of Web Services Atomic Transaction (WS-AT) and Web Services Business Activity (WS-BA) protocol messages across firewalls and outside the WebSphere Application Server domain. You configure an intermediary node to specify which WebSphere Application servers the node routes requests to.

Procedure

1. Configure the HTTP server so that it routes WS-AT and WS-BA requests that are targeted at WebSphere Application Server to WebSphere Application Server, rather than processing them itself.
2.
3.

Results

You configured the intermediary node ready for use by WebSphere Application Server.

Example

What to do next

Configure WebSphere Application Server to use the intermediary node by specifying, for each server, the appropriate virtual host of the intermediary node.

Example: Configuring IBM HTTP server as an intermediary node for web services transactions

You can use an HTTP server intermediary nodes to enable the exchange of Web Services Atomic Transaction and Web Services Business Activity protocol messages across firewalls and outside the WebSphere Application Server domain. For IBM HTTP server, you achieve this behavior by modifying the plugin-cfg.xml file of the IBM HTTP server node.

Routing requests to WebSphere Application Server

You can use the IBM HTTP server as a single intermediary node, or you can combine it with a Proxy Server for IBM WebSphere Application Server. In both cases, update the plugin-cfg.xml file to indicate that the HTTP server should route requests that are targeted at WebSphere Application Server, those of the form http://*host:port*/_IBMSYSAPP/*, to WebSphere Application Server, rather than processing them itself.

To update the plugin-cfg.xml file, add a URI element with a name of _IBMSYSAPP, as shown in the following example. Add this URI to all UriGroup elements in the plugin-cfg.xml file.

```
<UriGroup Name="default_host_server1_99T73NKNode01_Cluster_URIs">
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/snoop/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/hello" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/hitcount" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="*.jsp" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="*.jsv" />
```

```
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="*.jsw" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/j_security_check" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/ibm_security_logout" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/servlet/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/SamplesGallery/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/WSsamples/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/PlantsByWebSphere/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/PlantsByWebSphere/docs/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/_IBMSYSAPP/*" />
</UriGroup>
```

Configuring virtual host mapping

If you are using IBM HTTP as the only intermediary node, in other words you are not also using Proxy Server for IBM WebSphere Application Server, configure virtual hosts to represent each WebSphere Application Server that the HTTP node routes requests to. Update the plugin-cfg.xml file by adding VirtualHostGroup, VirtualHost and Route elements.

The following example shows part of the plugin-cfg.xml file for a configuration in which the IBM HTTP server routes requests to one of two servers, server1 and server2, in WebSphere Application Server.

The plugin-cfg.xml file contains two virtual host aliases, with names name1.acme.com and name2.acme.com, that are defined using VirtualHost and VirtualHostGroup elements. The Route elements define the association between the virtual hosts and the ServerCluster elements. When a request is made, IBM HTTP server finds the best matching route to dispatch the request to. A request made to virtual host name1.acme.com, with a URI that matches a pattern in the default_URIs URI group, is sent to the server1_Cluster server cluster. This server cluster contains only one server, server1, so requests targeted at virtual host name1.acme.com are sent to server1, and similarly, requests targeted at virtual host name2.acme.com are sent to server2.

```
<UriGroup Name="default_URIs">
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/snoop/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/hello" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/hitcount" />
...
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/PlantsByWebSphere/*" />
<Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/PlantsByWebSphere/docs/*" />
  <Uri AffinityCookie="JSESSIONID" AffinityURLIdentifier="jsessionid" Name="/_IBMSYSAPP/*" />
</UriGroup>

<ServerCluster CloneSeparatorChange="false" LoadBalance="Round Robin" Name="server1_Cluster" PostBufferSize="64"
PostSizeLimit="-1" RemoveSpecialHeaders="true" RetryInterval="60">
      <Server ConnectTimeout="0" ExtendedHandshake="false" MaxConnections="-1" Name="server1" ServerIOTimeout="0" WaitForContinue="false">
         ...
      </Server>
      <PrimaryServers> <Server Name="server1"/> </PrimaryServers>
   </ServerCluster>
<ServerCluster CloneSeparatorChange="false" LoadBalance="Round Robin" Name="server2_Cluster" PostBufferSize="64" PostSizeLimit="-1"
 RemoveSpecialHeaders="true" RetryInterval="60">
      <Server ConnectTimeout="0" ExtendedHandshake="false" MaxConnections="-1" Name="server2" ServerIOTimeout="0" WaitForContinue="false">
         ...
     </Server>
      <PrimaryServers> <Server Name="server2"/>  </PrimaryServers>
</ServerCluster>
<VirtualHostGroup Name="vhost_server1"> <VirtualHost Name="name1.acme.com:9081"/> </VirtualHostGroup>
<VirtualHostGroup Name="vhost_server2"> <VirtualHost Name="name2.acme.com:9081"/> </VirtualHostGroup>
<Route ServerCluster="server1_Cluster" UriGroup="default_URIs" VirtualHostGroup=" vhost_server1 "/>
<Route ServerCluster="server2_Cluster" UriGroup="default_URIs" VirtualHostGroup=" vhost_server2"/>
```

Enabling WebSphere Application Server to use an intermediary node for web services transactions

You can use intermediary nodes with Web Services Atomic Transactions (WS-AT) or Web Services Business Activities (WS-BA) to support the exchange of associated requests across firewalls and outside the WebSphere Application Server domain. You configure WebSphere Application Server to use an intermediary node by specifying the external endpoint URL information for the intermediary node in each server that is accessed through the intermediary.

Before you begin

Configure the intermediary node that you want to use, and ensure that you know the address or addresses of the intermediary node that you want to map to servers in your WebSphere Application Server configuration.

Configure the intermediary to listen on a specific port for protocol messages and to route these messages to the specific WebSphere Application Server instance that you want to enable. See the related information for an example configuration where an IBM HTTP server is the intermediary.

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Services] Transaction Service**.
2. In the External WS-Transaction HTTP(S) URL prefix section, click **Specify custom prefix**, then type the external endpoint URL information for the intermediary node in the field. Use one of the following formats for the prefix, where *host_name* and *port* represent the intermediary node that is an HTTP or HTTPS proxy for the server, and *port* is optional.
   - `http://host_name:port`
   - `https://host_name:port`
3. Click **Apply** or **OK**.
4. Save your changes to the master configuration.
5. Repeat the previous steps for each server that is accessed through the intermediary node.
6. Restart the servers.

Results

You configured your system to use an intermediary node. Test your configuration to ensure that messages are routed as you expect.

Configuring a server to use business activity support

Business activity support provides compensation for activities such as sending an email, which can be difficult or impossible to roll back atomically. With this compensation, applications on disparate systems can coordinate activities that are more loosely coupled than atomic transactions. To use the business activity support, you must first enable it on each server that you plan to use.

About this task

If an application component uses business activity support, you must enable the support on each server that runs the application.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

Procedure

1. In the administrative console, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **[Container Settings] Container Services** > **Compensation Service**.
2. Select the **Enable service at server startup** check box.

3. If required, modify the compensation handler retry interval and limit. These values control the frequency with which the compensation handler compensate and close methods are retried, when either throw a RetryCompensationHandlerException exception, and the number of times that these methods are retried.
4. Save your changes to the master configuration.
5. Repeat the previous steps for each server that you plan to use.
6. Restart all the servers for the changes to take effect.

Results

The business activity support is enabled for the application server. Verify a successful enablement by checking for the message, `CWSCP0005I: The Compensation service started successfully.` in the `SystemOut.log` file for the relevant server.

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

What to do next

Deploy the business-activity-enabled application to the server.

Chapter 38. Administering web services - Transports

Transport chains represent a network protocol stack that is used for I/O operations within an application server environment. Transport chains are part of the channel framework function that provides a common networking service for all components.

Invoking JAX-WS web services asynchronously using the HTTP transport

Using the JAX-WS asynchronous response servlet

Java API for XML-Based Web Services (JAX-WS) includes an asynchronous response servlet, which is used within the application server environment to receive responses for JAX-WS requests that are invoked asynchronously.

Before you begin

JAX-WS provides support for invoking web services using an asynchronous client invocation by using either a callback or polling model. Both the callback model and the polling model are available on the Dispatch client and the dynamic proxy client. When a JAX-WS client that is running within the application server environment uses an asynchronous client invocation, the responses are received by the asynchronous response servlet. To learn how to use the asynchronous client invocation model, read about invoking JAX-WS web services asynchronously.

About this task

The asynchronous response servlet is used within an application server to handle incoming asynchronous responses. The servlet uses the same secure and unsecure HTTP ports assigned to the application server. The servlet starts automatically when the application server starts. Because the asynchronous response servlet does not perform role-based authorization checks, only user authentication checks are performed.

The asynchronous response servlet supports both the HTTP and HTTPS protocols. Since the servlet inherits the SSL configuration of the application server, configuring the application server also configures the servlet. The asynchronous response servlet is not affected by the custom HTTP and SSL port properties used by the asynchronous response listener and only runs on the application ports for the application server.

Procedure

1. Determine if you want the JAX-WS client to use the HTTP or HTTPS transport mechanism.
2. Configure the web container transport chains to modify the SSL configuration of the application server. The servlet inherits these settings. Read about configuring transport chains to learn how to configure the web container transport chains.

Results

The asynchronous response servlet is configured to enable your JAX-WS clients to receive asynchronous responses on the HTTP or HTTPS transport protocol.

Note: When you add a new application server to your environment, the asynchronous response servlet is automatically restarted so the deployment.xml file can be updated for the new application server. If your application receives an incoming response when the asynchronous response servlet is restarting, the incoming response might fail with an HTTP 404 error.

Note: JAX-WS services do not successfully return asynchronous responses to clients that are installed in application security-enabled WebSphere Application Servers. Because the Asynchronous Response Servlet for WebSphere Application Server, which handles asynchronous web services responses, is protected when application security is enabled, you must supply a credential together with the JAX-WS service incoming response. Attach the HTTPTransport policy set binding to the JAX-WS service in the service attachment. Additionally, enter a valid basic authentication user ID and password, which are defined in the user registry of the client, into the Basic authentication for outbound asynchronous service responses field.

Using the JAX-WS asynchronous response listener

Java API for XML-Based Web Services (JAX-WS) includes an asynchronous response listener, which is used within the Thin Client for JAX-WS and application client environments to receive responses for requests that are invoked asynchronously.

Before you begin

JAX-WS provides support for invoking web services using an asynchronous client invocation by using either a callback or polling model. Both the callback model and the polling model are available on the Dispatch client and the dynamic proxy client. When the JAX-WS client uses an asynchronous client invocation, the responses are received by the asynchronous response listener. To learn how to use the asynchronous client invocation model, read about invoking JAX-WS web services asynchronously.

About this task

The asynchronous response listener is used within a Web services client to handle incoming asynchronous responses. You can use the listener in Thin Client for JAX-WS environments and application client environments. By default, the listener opens a random port to listen for asynchronous responses or you can optionally configure a specific port for the listener to use. The listener starts automatically in the JAX-WS run time when the JAX-WS client is configured to expect an asynchronous response.

There are two versions of the asynchronous response listener. The unsecure version of the asynchronous response listener supports the HTTP protocol, and the secure version of the asynchronous response listener supports the HTTPS protocol. The correct asynchronous response listener is automatically started based on the particular transport used by the JAX-WS client. To ensure that the correct Secure Sockets Layer (SSL) handshaking occurs between the asynchronous response listener and the application server, configure the SSL properties using the SSL transport policy or the Java system properties.

For web services clients running in the application server environment, use the asynchronous response servlet for receiving asynchronous responses.

Procedure

1. Determine if you want the JAX-WS client to use the HTTP or HTTPS transport mechanism.
2. Configure the asynchronous response listener for unsecure communication using HTTP.

   You can configure the HTTP port for the asynchronous response listener as a Java system property or as a custom property within the transport policy. Properties that are defined in the policy set binding files override any Java system property that might have been defined.

   a. Define the `com.ibm.websphere.webservices.http.listenerPort` property as a Java system property. If this property is set as a Java system property, then all asynchronous response listeners within that Java Virtual Machine (JVM) are affected.

   b. Define the `com.ibm.websphere.webservices.http.listenerPort` property within the `HTTPTransport` transport policy set bindings files. If this property is set as a custom property within a transport policy set binding, then only the services for which the policy set has been configured are affected.
3. Configure the asynchronous response listener for secure communication using HTTPS.

You can configure the HTTPS port for the asynchronous response listener as a Java system property or as a custom property within the transport policy.

a. Define the `com.ibm.websphere.webservices.https.listenerPort` property as a Java system property. If this property is set as a Java system property, thenall asynchronous response listeners within that JVM are affected.
b. Define the `com.ibm.websphere.webservices.https.listenerPort` property within the `SSLTransport` transport policy set bindings files. If this property is set as a custom property within a transport policy set binding, then only the services for which the policy set has been configured are affected.

Results

Your JAX-WS web services client is configured to use the asynchronous response listener to receive incoming asynchronous responses.

Example

The following examples demonstrate how to enable the asynchronous response listener when defining the custom port of 9999:

Use the following Java command to configure the custom HTTP port for the asynchronous response listener in a thin client environment:

```
- java.exe -Dcom.ibm.websphere.webservices.http.listenerPort=9999 com.ibm.websphere.my_program
```

Use the following launchClient command to configure the custom HTTP port for the asynchronous response listener in an application client container:

```
- launchClient.bat MyClient.ear -CCDcom.ibm.websphere.webservices.http.listenerPort=9999
```

The following is an excerpt from an HTTPTransport policy binding.xml file that includes the asynchronous response listener properties:

```
</wsp:Policy>
   </wsp:ExactlyOne>
 </wsp:All>
    <wshttp:outAsyncResponseProxy>
     <wshttp:connectInfo host="" port=""></wshttp:connectInfo>
                  <wshttp:basicAuth userid="" password=""></wshttp:basicAuth>
    </wshttp:outAsyncResponseProxy>
    <wshttp:properties>
  <wshttp:customProperty name="com.ibm.websphere.webservices.http.listenerPort" value="9999" />
   </wshttp:properties>
 </wsp:All>
   </wsp:ExactlyOne>
</wsp:Policy>
```

What to do next

Run the JAX-WS client with the specified asynchronous response listener options.

Invoking JAX-WS web services asynchronously using the SOAP over JMS transport

Using the JAX-WS JMS asynchronous response message listener

Java API for XML-Based Web Services (JAX-WS) includes a Java Message Service (JMS) asynchronous response message listener, which is used to receive responses to asynchronous JAX-WS requests that use the JMS transport. The JMS asynchronous response message listener is used in the application server and application client environments.

Before you begin

JAX-WS provides support for invoking web service operations asynchronously by using either a callback or a polling model. When the JAX-WS client uses the JMS transport to invoke asynchronous operations, the responses are received by the asynchronous response message listener. To learn how to use the JAX-WS asynchronous client invocation model, read about invoking JAX-WS web services asynchronously.

About this task

The JMS asynchronous response message listener is used within the web services client environment to receive incoming asynchronous responses when the client application is using the JMS Transport. The listener requires a connection factory and a queue to function correctly. Begin by configuring the connection factory and queue, and then specify the JNDI names of the connection factory and queue to the listener by setting Java system properties. The environment in which the client is running determines how the system properties are set.

The JMS asynchronous response message listener is started automatically by the web services client runtime environment when the client invokes the first asynchronous JAX-WS operation using the JMS transport.

The connection factory and the queue configured with the asynchronous response message listener is used for all requests that are invoked within a particular Java process such as for the application server or an application client container. You can share the connection factory among different Java processes. However, you cannot share a queue among Java processes.

Procedure

1. Determine if you want the JAX-WS client to use the JMS transport mechanism.
2. For each Java process that will use JMS as a transport for asynchronous JAX-WS requests, configure the connection factory and queue that are used by the JMS asynchronous response listener for that process. You can share a connection factory among multiple Java processes, but you cannot share a queue among Java processes.
3. For each Java process, set the com.ibm.websphere.webservices.jms.AsyncReplyQueueName and com.ibm.websphere.webservices.jms.AsyncReplyCFName Java system properties to specify the JNDI names of the queue and connection factory that are used by the JMS asynchronous response message listener for that process.

   If the JNDI name of the queue is the default value, jms/DefaultAsyncReplyQueue, then you do not need to set the AsyncReplyQueueName property. Likewise, if the JNDI name of the connection factory is the default value, jms/DefaultAsyncReplyCF, then you do not need to set the AsyncReplyCFName property as well.

   If your client runs within the application server environment, then set the properties as application server system properties by using the administrative console or the wsadmin command.

   If your client runs within the application client container environment, then you should set the properties by using the –CCD option on the launchClient command line.

Results

Your JAX-WS web services client is configured to use the JMS asynchronous response message listener to receive asynchronous response messages when using the JMS transport.

Example

Suppose that you have a JAX-WS web services client that runs in the application client container environment and uses the JMS transport to communicate with the server. Suppose also that the client invokes asynchronous JAX-WS operations. You can create a connection factory with the JNDI name,

jms/MyAppCF, and a queue with the JNDI name, jms/MyAppAsyncReplyQueue. When you invoke the client with the launchClient command, specify the JNDI names of the queue and connection factory as illustrated in the following command:

```
launchClient MyAppClient.ear \
   -CCDcom.ibm.websphere.webservices.jms.AsyncReplyQueueName=jms/MyAppReplyQueue \
   -CCDcom.ibm.websphere.webservices.jms.AsyncReplyCFName=jms/MyAppCF \
   <application arguments>
```

Chapter 39. Administering web services - UDDI registry

The Universal Description, Discovery, and Integration (UDDI) specification defines a way to publish and discover information about web services. The UDDI specification defines a standard for the visibility, reusability, and manageability that are essential for a service-oriented architecture (SOA) registry service. The UDDI registry is a directory for web services that is implemented using the UDDI specification. It is a component of WebSphere® Application Server.

Web services are self-contained, modular applications that can be described, published, located, and invoked over a network. They implement a services oriented architecture (SOA), which supports the connecting or sharing of resources and data in a very flexible and standardized manner. Services are described and organized to support their dynamic, automated discovery and reuse.

Administering the UDDI registry

You can set up and deploy a UDDI registry, then remove, reinstall, or apply an upgrade to a UDDI registry. You can configure SOAP API and GUI services for the UDDI registry. You can use the administrative console or the Java Management Extensions (JMX) management interface to manage UDDI registries.

Procedure

- Set up and deploy a new UDDI registry
- Remove a UDDI registry node
- Reinstall the UDDI registry application
- Apply an upgrade to the UDDI registry
- Configure SOAP API and GUI services for the UDDI registry
- Manage the UDDI registry

Setting up and deploying a new UDDI registry

A UDDI registry node consists of the UDDI registry application (an enterprise application that is supplied as part of WebSphere Application Server), a store of data (using a relational database management system) referred to as the UDDI database, and a means to connect the application to the data (a data source and related elements). To set up a new UDDI registry, you create the UDDI database and data source, and deploy the supplied application.

Before you begin

Start WebSphere Application Server, and create a server to host the UDDI registry. Use the starting and stopping quick reference information for information about starting WebSphere Application Server using either commands or the administrative console.

About this task

The subtopics describe how to create the UDDI database (which can be local or remote) and data source, and how to deploy the UDDI registry application.

You can create either a *default* UDDI node or a *customized* UDDI node. The main difference between the two nodes is the number of mandatory UDDI registry properties, such as the UDDI node ID and description, and the prefix to use for generated discovery URLs.

Default UDDI node

The mandatory properties are automatically set to default values and you cannot change them. A default UDDI node is a suitable option for initial evaluation of the UDDI registry, and for development and test purposes.

Customized UDDI node

You must set the mandatory properties. After these properties are set, you cannot change them for this configuration. With a customized UDDI node, you have more control over the database management system that is used for the UDDI database, and the properties that are used to set up the UDDI database. With a customized UDDI node, you create the UDDI database and data source to your own specifications before deploying the UDDI registry application. A customized node is a suitable option for production purposes. To move from a default UDDI node to a customized UDDI node, see “Changing the UDDI registry application environment after deployment” on page 3505.

Procedure

- To set up a UDDI registry quickly for test or development purposes, follow the instructions in “Setting up a default UDDI node with a default data source.” The database, data source, and UDDI registry application are created or deployed by a single script. The database type is embedded Apache Derby.
- To create a default UDDI registry with a database other than embedded Apache Derby, or to use an embedded Apache Derby database but create the data source manually, follow the instructions in “Setting up a default UDDI node” on page 3483.
- To create a customized UDDI registry, follow the instructions in “Setting up a customized UDDI node” on page 3493.

Setting up a default UDDI node with a default data source

You can create a UDDI node with predetermined property values and an embedded Apache Derby database. You can do this to set up a UDDI registry quickly for test or development purposes.

About this task

When you set up a default UDDI node with a default data source, the mandatory node properties, such as node ID, are set automatically and you cannot change them. You run a single script to create the UDDI database and data source, and to deploy the UDDI registry application. This type of node is suitable for initial evaluation of the UDDI registry and for development and test purposes.

If you want to create a default UDDI node with an embedded Apache Derby database but a different data source, or with a different database, see “Setting up a default UDDI node” on page 3483.

If you want to set up a UDDI node with your own properties, including the mandatory node properties, you must set up a customized node. See “Setting up a customized UDDI node” on page 3493.

Procedure

1. Create the UDDI node by running the wsadmin script uddiDeploy.jacl from the *app_server_root*/bin directory.

   Use the following syntax:

```
wsadmin [-conntype none] [-profileName profile_name]
              -wsadmin_classpath app_server_root/derby/lib
              -f uddiDeploy.jacl
              node_name
              server_name
              default
```

   where:

   - `-profileName` *profile_name* is optional, and is the name of the profile in which the UDDI application is deployed. If you do not specify a profile, the default profile is used.
   - `-conntype none` is optional, and is needed only if the application server is not running.
   - *app_server_root* is the directory name of the WebSphere Application Server installation location.

- *node_name* is the name of the WebSphere node on which the target server runs. Note that the node name is case sensitive.
- *server_name* is the name of the target server that deploys the UDDI registry, for example, server1. The server name is case sensitive.
- `default` creates a UDDI node with default policies in an Apache Derby database and data source. This option is specific to the Apache Derby database only and creates everything required to run a UDDI node.

  If the Apache Derby database already exists, you are asked if you want to re-create it. If you choose to re-create the database, the existing database is deleted and a new one is created in its place. If you choose not to re-create the database, the command exits and a new database is not created.

  Note: If the application server has already accessed the existing Apache Derby database, the uddiDeploy.jacl script cannot re-create the database. Use the uddiRemove.jacl script to remove the database, as described in the topic about removing a UDDI registry node, restart the server, and run the uddiDeploy.jacl script again.

For example, to create a UDDI node named MyNode on a server named server1 when server1 is started, you might enter the following command. Enter the command on a single line.

```
wsadmin -profileName myProfile -wsadmin_classpath /QIBM/ProdData/WebSphere/
AppServer/was_version/Base/derby/lib -f uddiDeploy.jacl MyNode server1 default
```

To create a UDDI node named MyNode on a server named server1 when server1 is not started, you might enter the following command. Enter the command on a single line.

```
wsadmin -conntype none -profileName myProfile -wsadmin_classpath /QIBM/ProdData/
WebSphere/AppServer/was_version/Base/derby/lib -f uddiDeploy.jacl MyNode server1 default
```

2. Click **Applications** > **Application Types** > **WebSphere enterprise applications** to display the installed applications. Start the UDDI registry application by selecting the check box next to the application name and clicking **Start**. Alternatively, if the application server is not already running, start the application server. This action automatically starts the UDDI registry application. The UDDI node is now active.

   If you restart the UDDI application or the application server, the UDDI node always reactivates, even if the node was previously deactivated.
3. Click **UDDI** > **UDDI Nodes** > *UDDI_node_id* to display the properties page for the UDDI registry node. Set **Prefix for generated discoveryURLs** to a valid URL for your configuration. This property specifies the URL prefix that is applied to generated discovery URLs that are used by the HTTP GET service for UDDI Version 2.

What to do next

Follow the instructions in “Using the UDDI registry installation verification test (IVT)” on page 3504 to verify that you have successfully set up the UDDI node.

Setting up a default UDDI node

You can create a UDDI node with predetermined property values. This UDDI node is suitable for initial evaluation of the UDDI registry and for development and test purposes.

About this task

When you set up a default UDDI node, the mandatory node properties, such as node ID, are set automatically and you cannot change them. You can create a default UDDI node with an embedded Apache Derby database but a different data source, or with a database other than Apache Derby.

If you want to set up a UDDI node with your own properties, including the mandatory node properties, you must set up a customized node. See “Setting up a customized UDDI node” on page 3493.

Procedure

1. Create a database schema to hold the UDDI registry by completing one of the following tasks, ensuring that you use the default node options where specified:
   - "Creating a DB2 distributed database for the UDDI registry"
   - "Creating a DB2 for i database for the UDDI registry" on page 3486
   - "Creating an Apache Derby database for the UDDI registry" on page 3487
   - "Creating an Oracle database for the UDDI registry" on page 3488
2. Set up a data source for the UDDI registry application to use to access the database, as described in "Creating a data source for the UDDI registry" on page 3489.
3. Deploy the UDDI registry application, as described in "Deploying the UDDI registry application" on page 3492.
4. Click **Applications** > **Application Types** > **WebSphere enterprise applications** to display the installed applications. Start the UDDI registry application by selecting the check box next to the application name and clicking **Start**. Alternatively, if the application server is not already running, start the application server. This action automatically starts the UDDI registry application. The UDDI node is now active.

   If you restart the UDDI application or the application server, the UDDI node always reactivates, even if the node was previously deactivated.
5. Click **UDDI** > **UDDI Nodes** > *UDDI_node_id* to display the properties page for the UDDI registry node. Set **Prefix for generated discoveryURLs** to a valid URL for your configuration. This property specifies the URL prefix that is applied to generated discovery URLs that are used by the HTTP GET service for UDDI Version 2.

What to do next

Because you chose to use a default UDDI node, the node is initialized when the UDDI application is started for the first time. Follow the instructions in "Using the UDDI registry installation verification test (IVT)" on page 3504 to verify that you have successfully set up the UDDI node.

Creating a DB2 distributed database for the UDDI registry:

Complete this task if you want to use DB2 on the Windows, Linux, or UNIX operating systems as the database store for your UDDI registry data.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

DataBaseName
: The name of the UDDI registry database. A suggested value is UDDI30. The UDDI information uses the suggested name of UDDI30, so if you use a different name, remember to substitute it when you see UDDI30 in the UDDI information.

DB2UserID
: A DB2 user ID with administrative privileges.

DB2Password
: The password for the DB2 user ID.

BufferPoolName
: The name of a buffer pool for the UDDI registry database to use. A suggested value is uddibp, but you can use any name because the buffer pool is created as part of this task.

TableSpaceName
: The name of a table space. A suggested value is uddits, but you can use any name.

TempTableSpaceName
The name of a temporary table space. A suggested value is udditstemp, but you can use any name because the temporary table space is created as part of this task.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Change the directory to *app_server_root*/UDDIReg/databaseScripts.
2. Start the DB2 Command Line Processor. Enter the following command at the command prompt:
   ```
   db2
   ```
3. Set up the DB2 environment variables. Enter the following command:
   ```
   set DB2CODEPAGE=1208
   ```
4. Create the DB2 database. Enter the following command:
   ```
   create database DataBaseName using codeset UTF-8 territory en
   ```
5. Configure the DB2 database. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   update db cfg for DataBaseName using applheapsz 2048
   ```
   c.
   ```
   update db cfg for DataBaseName using logfilsiz 8192
   ```
   d.
   ```
   connect reset
   ```
   e.
   ```
   terminate
   ```
   f.
   ```
   force application all
   ```
   g.
   ```
   terminate
   ```
   h.
   ```
   stop
   ```
   i.
   ```
   start
   ```
6. Restart the DB2 Command Line Processor. For all operating systems except Windows, enter the following command at the command prompt:
   ```
   db2
   ```
7. Create further database structures. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   create regular tablespace uddits pagesize 32K managed by system using
   ('TableSpaceName') extentsize 64 prefetchsize 32 bufferpool BufferPoolName
   ```
   c.
   ```
   create system temporary tablespace TempTableSpacename pagesize 32K managed by
   system using ('TempTableSpacename') extentsize 32 overhead 14.06
   prefetchsize 32 transferrate 0.33 bufferpool BufferPoolName
   ```
8. Define the database structures that are needed to store the UDDI data.

   Exit the DB2 Command Line Processor and enter the following commands exactly as shown. Note that one step uses `-vf` rather than `-tvf`.
   a.
   ```
   db2 -tvf uddi30crt_10_prereq_db2.sql
   ```

b.
```
db2 -tvf uddi30crt_20_tables_generic.sql
```
c.
```
db2 -tvf uddi30crt_25_tables_db2udb.sql
```
d.
```
db2 -tvf uddi30crt_30_constraints_generic.sql
```
e.
```
db2 -tvf uddi30crt_35_constraints_db2udb.sql
```
f.
```
db2 -tvf uddi30crt_40_views_generic.sql
```
g.
```
db2 -tvf uddi30crt_45_views_db2udb.sql
```
h.
```
db2 -vf uddi30crt_50_triggers_db2udb.sql
```
i.
```
db2 -tvf uddi30crt_60_insert_initial_static_data.sql
```

9. Optional: To use the database as a default UDDI node, enter the following command:
```
db2 -tvf uddi30crt_70_insert_default_database_indicator.sql
```
10. Issue the following commands:
```
connect reset
terminate
```
11. Issue the following commands:
```
connect reset
terminate
```

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating a DB2 for i database for the UDDI registry:

Complete this task if you want to use DB2 for i as the database store for your UDDI registry data.

Before you begin

The default names of the UDDI registry schema in the SQL scripts listed in the following topic are IBMUDI30 and IBMUDS30. These names are the recommended values and are assumed throughout the UDDI information. To use different names, modify the SQL files listed, then substitute the new names when IBMUDI30 and IBMUDS30 are used in the information center.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Use IBM Navigator for i to run SQL scripts.
   a. Open IBM Navigator for i.
   b. Expand **My Connections** > **iSeriesName** > **Databases**.
   c. Select **iSeriesName**.
   d. Right-click **Run SQL Scripts...**.

   A **Run SQL Scripts** window opens.
2. Open the IBM i DB2 SQL files.
   a. Map a network drive to the root directory of your IBM i server integrated file system.
   b. In Windows Explorer, expand the `WAS_HOME/UDDIReg/databaseScripts` directory.

c. Open the following SQL files with a text editor (for example, Windows Notepad):
   - uddi30crt_10_prereq_db2_iSeries.sql
   - uddi30crt_20_tables_generic_iSeries.sql
   - uddi30crt_25_tables_db2udb_iSeries.sql
   - uddi30crt_30_constraints_generic_iSeries.sql
   - uddi30crt_35_constraints_db2udb_iSeries.sql
   - uddi30crt_40_views_generic_iSeries.sql
   - uddi30crt_45_views_db2udb_iSeries.sql
   - uddi30crt_50_triggers_db2udb_iSeries.sql
   - uddi30crt_60_insert_initial_static_data_iSeries.sql

3. Copy the text to the **Run SQL Scripts** window.
   a. In the text editor of the file `uddi30crt_10_prereq_db2_iSeries.sql`, click **Edit** > **Select All**.
   b. Click **Edit** > **Copy**.
   c. In the **Run SQL Scripts** window, click **Edit** > **Paste**.
   d. Click **Run** > **All**.
   e. After the script completes running, select all the SQL text and delete it from the **Run SQL Scripts** window.
   f. Repeat the previous steps for all the SQL scripts listed in step 2.
4. Optional: If you want to use the database as a default UDDI node, complete the following steps:
   a. Open `uddi30crt_70_insert_default_database_indicator.sql` as described in step 2.
   b. Copy and run `uddi30crt_70_insert_default_database_indicator.sql` as described in step 3.

What to do next

Continue to set up and deploy your UDDI registry node.

Creating an Apache Derby database for the UDDI registry:

Complete this task to use an Apache Derby database as the database store for your UDDI registry. You can use an embedded or network Apache Derby database, and the database store can be local or remote.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

arg1 The path of the SQL files. On a standard installation, the path is *app_server_root*/UDDIReg/databasescripts.

arg2 The path to the location where you want to install the Apache Derby database.

For example, *profile_root*/databases/com.ibm.uddi.

arg3 The name of the Apache Derby database. A recommended value is UDDI30, and this name is assumed throughout the UDDI information. If you use another name, substitute that name when UDDI30 is used in the UDDI information.

arg4 An optional argument. Either use the value DEFAULT, or omit this argument. Specify DEFAULT to use the database as a default UDDI node. This argument is case sensitive.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Start a Qshell session. Enter the STRQSH command from the IBM i command line.
2. Create a UDDI Apache Derby database by using UDDIDerbyCreate.jar. Run the following Java -jar command from the *app_server_root*/UDDIReg/databaseScripts directory.

```
java -Djava.ext.dirs=app_server_root/derby/lib:app_server_root/java/jre/lib/ext -jar UDDIDerbyCreate.jar
arg1 arg2 arg3 arg4
```

   If the Apache Derby database already exists, you are asked if you want to re-create it. If you choose to re-create the database, the existing database is deleted and a new one is created in its place. If you choose not to re-create the database, the command exits and a new database is not created.

   Note: If the application server has already accessed the existing Apache Derby database, the uddiDeploy.jacl script cannot re-create the database. Use the uddiRemove.jacl script to remove the database, as described in the topic about removing a UDDI registry node, restart the server, and run the uddiDeploy.jacl script again.
3. If you are using a remote database, which requires network Apache Derby, or if you want to use network Apache Derby for other reasons, for example, to use Apache Derby with a cluster, configure the Apache Derby Network Server framework. For details, see the section about managing the Derby Network Server in the Derby Server and Administration Guide.

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating an Oracle database for the UDDI registry:

Complete this task if you want to use Oracle as the database store for your UDDI registry data.

Before you begin

This task creates three new schemas: ibmuddi, ibmudi30 and ibmuds30. You cannot complete this task if schemas with these names exist already.

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

OracleUserID
: The Oracle user ID to use to create the database.

OraclePassword
: The password for the Oracle user ID.

The Oracle database must be a remote database; you cannot create a local database. Refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

The supported versions of Oracle are Version 9i and Version 10g. Each version has the following restrictions:

Table 322. UDDI restrictions and Oracle versions. The table lists the Version 9i and Version 10g restrictions for different UDDI parameters.

	Version 9i restrictions		Version 10g restrictions	
discoveryURL (Business)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
accessPoint (bindingTemplate)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
instanceParms (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 8192 characters		
overviewURL (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 4096 characters		
Digital Signature	maximum 4000 bytes			

Procedure

1. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_10_prereq_oracle.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_20_tables_generic.sql
```
2. Run one of the following commands, depending on the version of Oracle.
   - For Oracle Version 9i:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle_pre10g.sql
```
   - For Oracle Version 10g and later:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle.sql
```
3. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_30_constraints_generic.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_35_constraints_oracle.sql
```
   c.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_40_views_generic.sql
```
   d.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_45_views_oracle.sql
```
   e.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_50_triggers_oracle.sql
```
   f.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_60_insert_initial_static_data.sql
```
4. Optional: To use the database as a default UDDI node, run the following command:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_70_insert_default_database_indicator.sql
```

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating a data source for the UDDI registry:

You create a data source so that the UDDI registry can use it to access the UDDI database.

Before you begin

You must have already created the database for the UDDI registry.

About this task

Complete this task as part of setting up and deploying a new UDDI registry. The UDDI registry uses the data source to access the UDDI database.

Procedure

1. Optional: For network Apache Derby, create a Java 2 Connector (J2C) authentication data entry. This step is not required for embedded Apache Derby.
   a. Click **Security** > **Global security** > **[Authentication] Java Authentication and Authorization Service** > **J2C authentication data**.
   b. Click **New** to create a new J2C authentication data entry.
   c. Enter the following details:

      Alias A suitable short name, for example `UDDIAlias`.

      Userid

      The database user ID, for example `db2admin` for DB2, or `IBMUDDI` for Oracle, which is used to read and write to the UDDI registry database. For network Apache Derby, the user ID can be any value.

      Password
      The password that is associated with the user ID specified previously. For network Apache Derby, the password can be any value.

      Description
      A description of the user ID.

      Click **Apply**, then save the changes to the master configuration.
2. Create a JDBC provider, if a suitable one does not already exist, by using the following table to determine the provider type and implementation type for your chosen database.

Table 323. Provider types and implementation types. The table lists the correct provider type and implementation type for each database.

Database	Provider type	Implementation type
DB2	DB2 UDB for iSeries (Native)	Connection pool data source
Oracle	Oracle JDBC Driver	Connection pool data source
Embedded Apache Derby	Derby JDBC Driver	Connection pool data source
Network Apache Derby	Derby Network Server JDBC Driver provider	Connection pool data source
Microsoft SQL Server	DataDirect Connect JDBC Driver Microsoft SQL Server JDBC Driver	Connection pool data source

   For details about how to create a JDBC provider, see the topic about configuring a JDBC provider by using the administrative console.
3. Create the data source for the UDDI registry:
   a. Click **Resources** > **JDBC** > **JDBC Providers**.
   b. Select the scope of the JDBC provider that you selected or created earlier, that is, the level at which the JDBC provider is defined. For example, for a JDBC provider that is defined at the level of server1, select the following:

```
Node=Node01, Server=server1
```

      All the JDBC providers that are defined at the selected scope are displayed.
   c. Select the JDBC provider that you created earlier.
   d. Under **Additional Properties**, select **Data sources**. Do not select the **Data sources (WebSphere Application Server V4)** option.

e. Click **New** to create a new data source.

f. In the **Create a data source** wizard, enter the following data:

Name A suitable name, for example `UDDI Datasource`.

JNDI name
Enter `datasources/uddids`. This is a mandatory field.

You must not have any other data sources that use this Java Naming and Directory Interface (JNDI) name. If another data source uses this JNDI name, you must either remove it or change its JNDI name. For example, if you created a default UDDI node previously that uses an Apache Derby database, before you continue, use the uddiRemove.jacl script with the default option to remove the data source and the UDDI application instance.

Component-managed authentication alias

- For DB2, Oracle, or network Apache Derby, select the alias that you created in step 2. The alias is prefixed by the node name, for example MyNode/UDDIAlias.
- For embedded Apache Derby, select `(none)`.

g. Click **Next**.

h. On the database-specific properties page of the wizard, enter the following data:

- For DB2:

  Database name
  The name of the database, for example `*LOCAL`.

- For Oracle:

  URL The Uniform Resource Locator (URL) of the database from which the datasource obtains connections, for example `jdbc:oracle:oci8:@`*`Oracle_database_name`*.

- For Apache Derby (embedded or network):

  Database name
  The name of the database, for example:

profile_root`/databases/com.ibm.uddi/UDDI30`

For network Apache Derby, ensure that the **Server name** and **Port number** values match the network server.

Leave all other fields unchanged.

Use this Data Source in container-managed persistence (CMP)
Ensure that the check box is cleared.

i. Click **Next**, then check the summary and click **Finish**.

j. Click the data source to display its properties, and add the following information:

Description
A description of the data source.

Category
Enter `uddi`.

Data store helper class name
This value is provided automatically:

Table 324. Data store helper class names

Database	Data store helper class name
DB2	com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper
Oracle 11g	com.ibm.websphere.rsadapter.Oracle11gDataStoreHelper
Embedded Apache Derby	com.ibm.websphere.rsadapter.DerbyDataStoreHelper

Table 324. Data store helper class names (continued)

Database	Data store helper class name
Network Apache Derby	com.ibm.websphere.rsadapter.DerbyNetworkServerDataStoreHelper

Mapping-configuration alias
Select `DefaultPrincipalMapping`.

k. Click **Apply**.

l. Select **Additional Properties** > **Custom Properties** > **libraries**.

m. Enter `IBMUDI30,IBMUDS30` in the **Value** field and click **OK**.

n. Save the changes to the master configuration.

4. Test the connection to your UDDI database by selecting the check box next to the data source and clicking **Test connection**. A message similar to `Test Connection for datasource UDDI Datasource on server server1 at node Node01 was successful` is displayed. If a different message is displayed, use the information in that message to investigate and resolve the problem.

What to do next

Continue with setting up and deploying your UDDI registry node.

Deploying the UDDI registry application:

You deploy a UDDI registry application as part of setting up a UDDI node. You can use a supplied script, the administrative console, or wsadmin scripting commands.

Before you begin

Before you deploy a UDDI registry application, you must create the database and data source for the UDDI registry.

About this task

Use this task as part of setting up a default UDDI node or setting up a customized UDDI node. You can deploy a UDDI registry application in two ways:

- You can use a script that performs all the necessary steps.

  This script deploys the UDDI registry to a server that you specify.
- You can use the administrative console. You deploy the UDDI registry application, the `uddi.ear` file, then complete additional steps, as described later in this topic. Alternatively, you can follow the same procedure using wsadmin scripting commands.

Procedure

1. Optional: To deploy a UDDI registry application using the supplied script:

   a. Start a Qshell session by entering the STRQSH command from the IBM i command line.

   b. Run the `uddiDeploy.jacl` wsadmin script as shown, from the *app_server_root*/bin directory.

```
wsadmin [-conntype none] [-profileName profile_name] -f uddiDeploy.jacl
        node_name server_name
```

The attributes of the command are as follows:

- `-conntype none` is optional, and is needed only if the application server is not running.
- `-profileName` *profile_name* is optional, and is the name of the profile in which the UDDI application is deployed. If you do not specify a profile, the default profile is used.
- *node_name* is the name of the WebSphere Application Server node on which the target server runs. The node name is case sensitive.

- *server_name* is the name of the target server on which you want to deploy the UDDI registry, for example, server1. The server name is case sensitive.

For example, to deploy UDDI on the node MyNode and the server server1, assuming that server1 is already started:

```
wsadmin -f uddiDeploy.jacl MyNode server1
```

2. Optional: To deploy a UDDI registry application using the administrative console, use the following steps.
   a. Install the UDDI application (the `uddi.ear` file) to the server that you require.
   b. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***uddi_application*** > **[Detail Properties] Class loading and update detection**.
   c. Ensure that **Class loader order** is set to **Classes loaded with local class loader first (parent last)**.
   d. Ensure that **WAR class loader policy** is set to **Single class loader for application**.
   e. Click **Apply**, then save your changes to the master configuration.

What to do next

Continue setting up the UDDI node.

Setting up a customized UDDI node

You can set up a UDDI node with your own properties, including the mandatory node properties. This type of UDDI node is suitable for production purposes.

About this task

You can set up a customized UDDI node with property values that you choose. After the node is initialized, you cannot change the mandatory node properties, for example, the node ID.

Procedure

1. Review the information in Databases and production use of the UDDI registry to decide which database system to use, then create a database schema to hold the UDDI registry by completing one of the following tasks. Do not use the default node options where specified.
   - “Creating a DB2 distributed database for the UDDI registry” on page 3484
   - “Creating a DB2 for i database for the UDDI registry” on page 3486
   - “Creating an Apache Derby database for the UDDI registry” on page 3487
   - “Creating an Oracle database for the UDDI registry” on page 3488
2. Set up a data source for the UDDI registry application to use to access the database, as described in “Creating a data source for the UDDI registry” on page 3489.
3. Deploy the UDDI registry application, as described in “Deploying the UDDI registry application” on page 3492.
4. Click **Applications** > **Application Types** > **WebSphere enterprise applications** to display the installed applications. Start the UDDI registry application by selecting the check box next to the application name and clicking **Start**. Alternatively, if the application server is not already running, start the application server. This action automatically starts the UDDI registry application. The UDDI node is now active.

   If you restart the UDDI application or the application server, the UDDI node always reactivates, even if the node was previously deactivated.

What to do next

Because you chose a user-customized UDDI node, you must set the properties for the UDDI node by using UDDI administration, and initialize the node, before it is ready to accept UDDI requests. See

“Initializing the UDDI registry node” on page 3502 for details.

Creating a DB2 distributed database for the UDDI registry:

Complete this task if you want to use DB2 on the Windows, Linux, or UNIX operating systems as the database store for your UDDI registry data.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

DataBaseName
The name of the UDDI registry database. A suggested value is UDDI30. The UDDI information uses the suggested name of UDDI30, so if you use a different name, remember to substitute it when you see UDDI30 in the UDDI information.

DB2UserID
A DB2 user ID with administrative privileges.

DB2Password
The password for the DB2 user ID.

BufferPoolName
The name of a buffer pool for the UDDI registry database to use. A suggested value is uddibp, but you can use any name because the buffer pool is created as part of this task.

TableSpaceName
The name of a table space. A suggested value is uddits, but you can use any name.

TempTableSpaceName
The name of a temporary table space. A suggested value is udditstemp, but you can use any name because the temporary table space is created as part of this task.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Change the directory to *app_server_root*/UDDIReg/databaseScripts.
2. Start the DB2 Command Line Processor. Enter the following command at the command prompt:
   ```
   db2
   ```
3. Set up the DB2 environment variables. Enter the following command:
   ```
   set DB2CODEPAGE=1208
   ```
4. Create the DB2 database. Enter the following command:
   ```
   create database DataBaseName using codeset UTF-8 territory en
   ```
5. Configure the DB2 database. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   update db cfg for DataBaseName using applheapsz 2048
   ```
   c.
   ```
   update db cfg for DataBaseName using logfilsiz 8192
   ```

d.
```
connect reset
```
e.
```
terminate
```
f.
```
force application all
```
g.
```
terminate
```
h.
```
stop
```
i.
```
start
```

6. Restart the DB2 Command Line Processor. For all operating systems except Windows, enter the following command at the command prompt:
```
db2
```
7. Create further database structures. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   create regular tablespace uddits pagesize 32K managed by system using
   ('TableSpaceName') extentsize 64 prefetchsize 32 bufferpool BufferPoolName
   ```
   c.
   ```
   create system temporary tablespace TempTableSpacename pagesize 32K managed by
   system using ('TempTableSpacename') extentsize 32 overhead 14.06
   prefetchsize 32 transferrate 0.33 bufferpool BufferPoolName
   ```
8. Define the database structures that are needed to store the UDDI data.

   Exit the DB2 Command Line Processor and enter the following commands exactly as shown. Note that one step uses `-vf` rather than `-tvf`.
   a.
   ```
   db2 -tvf uddi30crt_10_prereq_db2.sql
   ```
   b.
   ```
   db2 -tvf uddi30crt_20_tables_generic.sql
   ```
   c.
   ```
   db2 -tvf uddi30crt_25_tables_db2udb.sql
   ```
   d.
   ```
   db2 -tvf uddi30crt_30_constraints_generic.sql
   ```
   e.
   ```
   db2 -tvf uddi30crt_35_constraints_db2udb.sql
   ```
   f.
   ```
   db2 -tvf uddi30crt_40_views_generic.sql
   ```
   g.
   ```
   db2 -tvf uddi30crt_45_views_db2udb.sql
   ```
   h.
   ```
   db2 -vf uddi30crt_50_triggers_db2udb.sql
   ```
   i.
   ```
   db2 -tvf uddi30crt_60_insert_initial_static_data.sql
   ```
9. Optional: To use the database as a default UDDI node, enter the following command:
```
db2 -tvf uddi30crt_70_insert_default_database_indicator.sql
```
10. Issue the following commands:
```
connect reset
terminate
```
11. Issue the following commands:
```
connect reset
terminate
```

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating a DB2 for i database for the UDDI registry:

Complete this task if you want to use DB2 for i as the database store for your UDDI registry data.

Before you begin

The default names of the UDDI registry schema in the SQL scripts listed in the following topic are IBMUDI30 and IBMUDS30. These names are the recommended values and are assumed throughout the UDDI information. To use different names, modify the SQL files listed, then substitute the new names when IBMUDI30 and IBMUDS30 are used in the information center.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Use IBM Navigator for i to run SQL scripts.
   a. Open IBM Navigator for i.
   b. Expand **My Connections** > **iSeriesName** > **Databases**.
   c. Select **iSeriesName**.
   d. Right-click **Run SQL Scripts...**.

   A **Run SQL Scripts** window opens.
2. Open the IBM i DB2 SQL files.
   a. Map a network drive to the root directory of your IBM i server integrated file system.
   b. In Windows Explorer, expand the `WAS_HOME/UDDIReg/databaseScripts` directory.
   c. Open the following SQL files with a text editor (for example, Windows Notepad):
      - uddi30crt_10_prereq_db2_iSeries.sql
      - uddi30crt_20_tables_generic_iSeries.sql
      - uddi30crt_25_tables_db2udb_iSeries.sql
      - uddi30crt_30_constraints_generic_iSeries.sql
      - uddi30crt_35_constraints_db2udb_iSeries.sql
      - uddi30crt_40_views_generic_iSeries.sql
      - uddi30crt_45_views_db2udb_iSeries.sql
      - uddi30crt_50_triggers_db2udb_iSeries.sql
      - uddi30crt_60_insert_initial_static_data_iSeries.sql
3. Copy the text to the **Run SQL Scripts** window.
   a. In the text editor of the file `uddi30crt_10_prereq_db2_iSeries.sql`, click **Edit** > **Select All**.
   b. Click **Edit** > **Copy**.
   c. In the **Run SQL Scripts** window, click **Edit** > **Paste**.
   d. Click **Run** > **All**.
   e. After the script completes running, select all the SQL text and delete it from the **Run SQL Scripts** window.
   f. Repeat the previous steps for all the SQL scripts listed in step 2.
4. Optional: If you want to use the database as a default UDDI node, complete the following steps:

a. Open `uddi30crt_70_insert_default_database_indicator.sql` as described in step 2.
b. Copy and run `uddi30crt_70_insert_default_database_indicator.sql` as described in step 3.

What to do next

Continue to set up and deploy your UDDI registry node.

Creating an Apache Derby database for the UDDI registry:

Complete this task to use an Apache Derby database as the database store for your UDDI registry. You can use an embedded or network Apache Derby database, and the database store can be local or remote.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

arg1 The path of the SQL files. On a standard installation, the path is *app_server_root*/UDDIReg/databasescripts.

arg2 The path to the location where you want to install the Apache Derby database.

For example, *profile_root*/databases/com.ibm.uddi.

arg3 The name of the Apache Derby database. A recommended value is UDDI30, and this name is assumed throughout the UDDI information. If you use another name, substitute that name when UDDI30 is used in the UDDI information.

arg4 An optional argument. Either use the value DEFAULT, or omit this argument. Specify DEFAULT to use the database as a default UDDI node. This argument is case sensitive.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Start a Qshell session. Enter the STRQSH command from the IBM i command line.
2. Create a UDDI Apache Derby database by using UDDIDerbyCreate.jar. Run the following Java -jar command from the *app_server_root*/UDDIReg/databaseScripts directory.

```
java -Djava.ext.dirs=app_server_root/derby/lib:app_server_root/java/jre/lib/ext -jar UDDIDerbyCreate.jar
arg1 arg2 arg3 arg4
```

   If the Apache Derby database already exists, you are asked if you want to re-create it. If you choose to re-create the database, the existing database is deleted and a new one is created in its place. If you choose not to re-create the database, the command exits and a new database is not created.

   Note: If the application server has already accessed the existing Apache Derby database, the uddiDeploy.jacl script cannot re-create the database. Use the uddiRemove.jacl script to remove the database, as described in the topic about removing a UDDI registry node, restart the server, and run the uddiDeploy.jacl script again.
3. If you are using a remote database, which requires network Apache Derby, or if you want to use network Apache Derby for other reasons, for example, to use Apache Derby with a cluster, configure the Apache Derby Network Server framework. For details, see the section about managing the Derby Network Server in the Derby Server and Administration Guide.

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating an Oracle database for the UDDI registry:

Complete this task if you want to use Oracle as the database store for your UDDI registry data.

Before you begin

This task creates three new schemas: ibmuddi, ibmudi30 and ibmuds30. You cannot complete this task if schemas with these names exist already.

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

OracleUserID
The Oracle user ID to use to create the database.

OraclePassword
The password for the Oracle user ID.

The Oracle database must be a remote database; you cannot create a local database. Refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

The supported versions of Oracle are Version 9i and Version 10g. Each version has the following restrictions:

Table 325. UDDI restrictions and Oracle versions. The table lists the Version 9i and Version 10g restrictions for different UDDI parameters.

	Version 9i restrictions		**Version 10g restrictions**	
discoveryURL (Business)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
accessPoint (bindingTemplate)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
instanceParms (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 8192 characters		
overviewURL (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 4096 characters		
Digital Signature	maximum 4000 bytes			

Procedure

1. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_10_prereq_oracle.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_20_tables_generic.sql
```
2. Run one of the following commands, depending on the version of Oracle.
   - For Oracle Version 9i:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle_pre10g.sql
```
   - For Oracle Version 10g and later:

```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle.sql
```

3. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_30_constraints_generic.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_35_constraints_oracle.sql
```
   c.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_40_views_generic.sql
```
   d.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_45_views_oracle.sql
```
   e.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_50_triggers_oracle.sql
```
   f.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_60_insert_initial_static_data.sql
```
4. Optional: To use the database as a default UDDI node, run the following command:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_70_insert_default_database_indicator.sql
```

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating a data source for the UDDI registry:

You create a data source so that the UDDI registry can use it to access the UDDI database.

Before you begin

You must have already created the database for the UDDI registry.

About this task

Complete this task as part of setting up and deploying a new UDDI registry. The UDDI registry uses the data source to access the UDDI database.

Procedure

1. Optional: For network Apache Derby, create a Java 2 Connector (J2C) authentication data entry. This step is not required for embedded Apache Derby.
   a. Click **Security** > **Global security** > **[Authentication] Java Authentication and Authorization Service** > **J2C authentication data**.
   b. Click **New** to create a new J2C authentication data entry.
   c. Enter the following details:

      Alias A suitable short name, for example `UDDIAlias`.

      Userid

      The database user ID, for example `db2admin` for DB2, or `IBMUDDI` for Oracle, which is used to read and write to the UDDI registry database. For network Apache Derby, the user ID can be any value.

      Password
      The password that is associated with the user ID specified previously. For network Apache Derby, the password can be any value.

      Description
      A description of the user ID.

      Click **Apply**, then save the changes to the master configuration.

2. Create a JDBC provider, if a suitable one does not already exist, by using the following table to determine the provider type and implementation type for your chosen database.

Table 326. Provider types and implementation types. The table lists the correct provider type and implementation type for each database.

Database	Provider type	Implementation type
DB2	DB2 UDB for iSeries (Native)	Connection pool data source
Oracle	Oracle JDBC Driver	Connection pool data source
Embedded Apache Derby	Derby JDBC Driver	Connection pool data source
Network Apache Derby	Derby Network Server JDBC Driver provider	Connection pool data source
Microsoft SQL Server	DataDirect Connect JDBC Driver Microsoft SQL Server JDBC Driver	Connection pool data source

   For details about how to create a JDBC provider, see the topic about configuring a JDBC provider by using the administrative console.
3. Create the data source for the UDDI registry:
   a. Click **Resources** > **JDBC** > **JDBC Providers**.
   b. Select the scope of the JDBC provider that you selected or created earlier, that is, the level at which the JDBC provider is defined. For example, for a JDBC provider that is defined at the level of server1, select the following:

```
Node=Node01, Server=server1
```

      All the JDBC providers that are defined at the selected scope are displayed.
   c. Select the JDBC provider that you created earlier.
   d. Under **Additional Properties**, select **Data sources**. Do not select the **Data sources (WebSphere Application Server V4)** option.
   e. Click **New** to create a new data source.
   f. In the **Create a data source** wizard, enter the following data:

      Name A suitable name, for example `UDDI Datasource`.

      JNDI name
      Enter `datasources/uddids`. This is a mandatory field.

      You must not have any other data sources that use this Java Naming and Directory Interface (JNDI) name. If another data source uses this JNDI name, you must either remove it or change its JNDI name. For example, if you created a default UDDI node previously that uses an Apache Derby database, before you continue, use the uddiRemove.jacl script with the default option to remove the data source and the UDDI application instance.

      Component-managed authentication alias
      - For DB2, Oracle, or network Apache Derby, select the alias that you created in step 2. The alias is prefixed by the node name, for example MyNode/UDDIAlias.
      - For embedded Apache Derby, select `(none)`.

   g. Click **Next**.
   h. On the database-specific properties page of the wizard, enter the following data:
      - For DB2:

        Database name
        The name of the database, for example `*LOCAL`.
      - For Oracle:

URL The Uniform Resource Locator (URL) of the database from which the datasource obtains connections, for example `jdbc:oracle:oci8:@Oracle_database_name`.

- For Apache Derby (embedded or network):

  Database name
  The name of the database, for example:

`profile_root/databases/com.ibm.uddi/UDDI30`

  For network Apache Derby, ensure that the **Server name** and **Port number** values match the network server.

  Leave all other fields unchanged.

Use this Data Source in container-managed persistence (CMP)
Ensure that the check box is cleared.

i. Click **Next**, then check the summary and click **Finish**.

j. Click the data source to display its properties, and add the following information:

Description
A description of the data source.

Category
Enter `uddi`.

Data store helper class name
This value is provided automatically:

Table 327. Data store helper class names

Database	Data store helper class name
DB2	com.ibm.websphere.rsadapter.DB2AS400DataStoreHelper
Oracle 11g	com.ibm.websphere.rsadapter.Oracle11gDataStoreHelper
Embedded Apache Derby	com.ibm.websphere.rsadapter.DerbyDataStoreHelper
Network Apache Derby	com.ibm.websphere.rsadapter.DerbyNetworkServerDataStoreHelper

Mapping-configuration alias
Select `DefaultPrincipalMapping`.

k. Click **Apply**.

l. Select **Additional Properties** > **Custom Properties** > **libraries**.

m. Enter `IBMUDI30,IBMUDS30` in the **Value** field and click **OK**.

n. Save the changes to the master configuration.

4. Test the connection to your UDDI database by selecting the check box next to the data source and clicking **Test connection**. A message similar to `Test Connection for datasource UDDI Datasource on server server1 at node Node01 was successful` is displayed. If a different message is displayed, use the information in that message to investigate and resolve the problem.

What to do next

Continue with setting up and deploying your UDDI registry node.

Deploying the UDDI registry application:

You deploy a UDDI registry application as part of setting up a UDDI node. You can use a supplied script, the administrative console, or wsadmin scripting commands.

Before you begin

Before you deploy a UDDI registry application, you must create the database and data source for the UDDI registry.

About this task

Use this task as part of setting up a default UDDI node or setting up a customized UDDI node. You can deploy a UDDI registry application in two ways:

- You can use a script that performs all the necessary steps.

  This script deploys the UDDI registry to a server that you specify.
- You can use the administrative console. You deploy the UDDI registry application, the `uddi.ear` file, then complete additional steps, as described later in this topic. Alternatively, you can follow the same procedure using wsadmin scripting commands.

Procedure

1. Optional: To deploy a UDDI registry application using the supplied script:
   a. Start a Qshell session by entering the STRQSH command from the IBM i command line.
   b. Run the `uddiDeploy.jacl` wsadmin script as shown, from the *app_server_root*/bin directory.

```
wsadmin [-conntype none] [-profileName profile_name] -f uddiDeploy.jacl
        node_name server_name
```

   The attributes of the command are as follows:
   - `-conntype none` is optional, and is needed only if the application server is not running.
   - `-profileName` *profile_name* is optional, and is the name of the profile in which the UDDI application is deployed. If you do not specify a profile, the default profile is used.
   - *node_name* is the name of the WebSphere Application Server node on which the target server runs. The node name is case sensitive.
   - *server_name* is the name of the target server on which you want to deploy the UDDI registry, for example, server1. The server name is case sensitive.

   For example, to deploy UDDI on the node MyNode and the server server1, assuming that server1 is already started:

```
wsadmin -f uddiDeploy.jacl MyNode server1
```

2. Optional: To deploy a UDDI registry application using the administrative console, use the following steps.
   a. Install the UDDI application (the `uddi.ear` file) to the server that you require.
   b. Click **Applications** > **Application Types** > **WebSphere enterprise applications** > ***uddi_application*** > **[Detail Properties] Class loading and update detection**.
   c. Ensure that **Class loader order** is set to **Classes loaded with local class loader first (parent last)**.
   d. Ensure that **WAR class loader policy** is set to **Single class loader for application**.
   e. Click **Apply**, then save your changes to the master configuration.

What to do next

Continue setting up the UDDI node.

Initializing the UDDI registry node:

Use this topic to initialize a UDDI registry node after setup or migration.

Before you begin

You must have a UDDI registry node set up, either as a new node, or to use for migrating a UDDI registry Version 2 node.

About this task

You must set some of the properties of a UDDI registry node before you initialize the node. The UDDI registry node properties are in two categories:

- Mandatory node properties that must be set before the UDDI node can be initialized. You can set these properties as many times as you want before initialization. However, after the UDDI node is initialized, these properties are read-only for the lifetime of that UDDI node. It is very important to set these properties correctly.
- All other properties. You can set these properties before and after initialization.

You can configure these properties and initialize the node by using the UDDI administrative console or Java Management Extensions (JMX) management interface.

Procedure

1. Click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** to display the properties page for the UDDI registry node.
2. Set the mandatory node properties to suitable, and valid, values. These properties are indicated by the presence of an asterisk (*) next to the input field. The following list summarizes the properties. For more information on each property, see the context help of the administrative console.

   UDDI node ID
   This property must be a text string that begins with `uddi:` and that is unique to this UDDI node. The default value might be suitable, but ensure that it is unique to the UDDI node before you use it.

   UDDI node description
   This property is a text string that describes the node.

   Root key generator
   This property must be a text string that begins with `uddi:` and that is unique to this UDDI node. The default value might be suitable, but might contain text, such as `keyspace_id`, that you must modify to match your system. If you use the default value, ensure that it is unique to this UDDI node.

   Prefix for generated discoveryURLs
   This property must be a valid Uniform Resource Locator (URL).
3. If you are migrating from Version 2 of the UDDI registry, use the following table to complete the following steps:
   - Set any properties from the `uddi.properties` file that must remain the same as Version 2.
   - Set any properties from the `uddi.properties` file that you want to keep the same value, for example, dbMaxResultCount.

Table 328. Version 2 and Version 3 UDDI properties. The table lists the different UDDI properties for Version 2 and Version 3, along with additional information about each one.

Version 2 UDDI property (which is set in uddi.property file)	Version 3 UDDI property (which is set by using the administrative console or UDDI Administrative Interface)	Recommended Version 3 UDDI property setting
dbMaxResultCount	Maximum inquiry response set size	You can retain the value from Version 2, change the value, or use the default.
persister	No equivalent	Not applicable.

Table 328. Version 2 and Version 3 UDDI properties (continued). The table lists the different UDDI properties for Version 2 and Version 3, along with additional information about each one.

Version 2 UDDI property (which is set in uddi.property file)	Version 3 UDDI property (which is set by using the administrative console or UDDI Administrative Interface)	Recommended Version 3 UDDI property setting
defaultLanguage	Default language code	Retain the value from Version 2
operatorName	UDDI node ID	You must use a valid value for the UDDI node ID. This value is applied to your Version 2 data as it is migrated.
maxSearchKeys	Maximum search keys	You can retain the value from Version 2, change the value, or use the default.
getServletURLprefix	Prefix for generated discoveryURLs	Enter a valid value for your configuration, which is the same as the value used for Version 2.
getServletName	No equivalent	Not applicable.

4. Set any other properties, such as policy values, that you want to change from the default settings. For an explanation of policies and properties, see the topic about UDDI node settings. Remember that you can also change these properties later.
5. Click **Apply** to save the changes.

   Important: Ensure that the mandatory node properties are set to appropriate values and that you have saved them, because you cannot change them after initialization. If you do not save your changes before proceeding to the initialize step, you will have to delete and recreate the database.
6. After saving the changes, initialize the UDDI node by clicking **Initialize**, at the top of the pane.

   If you are migrating from Version 2 of the UDDI registry, the Version 2 data is migrated now. The initialization might take some time to complete; to track its progress, return to the node collection page and click the refresh icon at the top of the **Status** column. Alternatively, open a second administrative console window, and use the refresh icon in the same way. The UDDI node goes through the following states:
   a. Initialization pending.
   b. Initialization in progress.
   c. Migration in progress. This state occurs only if you are migrating.
   d. Value set creation in progress.
   e. Activated.

What to do next

If you migrated the node from a previous version, return to the steps in the procedure to migrate to Version 3 of the UDDI registry, to verify that the migration was successful. If you created a new node, follow the steps in “Using the UDDI registry installation verification test (IVT)” to verify that you have successfully set up the UDDI node.

Using the UDDI registry installation verification test (IVT)

You can use an installation verification test (IVT) to verify that you have successfully deployed a UDDI registry.

Before you begin

You must have set up and deployed a new UDDI registry.

Procedure

1. Open a browser window and enter the URL that accesses the UDDI registry user interface.

2. Under the **Quick Find** heading on the **Find** tab, select the **Business** radio button and enter a percent symbol (%) in the **Starting with** field.
3. Click **Find**. If you deployed your UDDI registry successfully, the detail frame shows the business entity that represents this UDDI node. You can click on the business entity to see its detail.

What to do next

As a further installation verification test, you can publish and find more UDDI entities by using the UDDI registry user interface. Alternatively, you can compile and run one or more of the UDDI registry samples that are available.

Changing the UDDI registry application environment after deployment

You can change the environment of the UDDI registry application after you deploy it. For example, you can evaluate a UDDI registry using one database, then put it into production using a different database.

About this task

After you deploy a UDDI registry application, you might want to change its environment. For example, you might complete initial evaluation of the UDDI registry by using an Apache Derby database, and then put the UDDI registry into production by using a DB2 database.

Procedure

1. Optional: To move from a default UDDI node to a customized UDDI node, delete the UDDI registry database and recreate it by completing one of the following tasks, ensuring that you do not use the default node options where specified:
   - “Creating a DB2 distributed database for the UDDI registry” on page 3484
   - “Creating a DB2 for i database for the UDDI registry” on page 3486
   - “Creating an Apache Derby database for the UDDI registry” on page 3487
   - “Creating an Oracle database for the UDDI registry” on page 3488

   Note: Any data that is saved in the default node (policies, properties, and user data) is lost when you delete the database. If you do not want to delete the database, create an entirely new customized UDDI node in a separate application server. The default UDDI node still exists for you to use for test purposes.
2. Optional: To change the database type for the UDDI registry, complete the following steps:
   a. Stop the UDDI registry application. Click **Applications** > **Application Types** > **WebSphere enterprise applications**, select the relevant check box, then click **Stop**.
   b. Either change the Java Naming and Directory Interface (JNDI) name of the existing data source from `datasources/uddids` to another value, or delete the data source. To display the data source properties, click **Resources** > **JDBC** > **JDBC providers** > ***database_type*** **JDBC Provider** > **[Additional Properties] Data sources** > ***uddi_datasource***.
   c. Create the new database by referring to one of the following topics:
      - “Creating a DB2 distributed database for the UDDI registry” on page 3484
      - “Creating a DB2 for i database for the UDDI registry” on page 3486
      - “Creating an Apache Derby database for the UDDI registry” on page 3487
      - “Creating an Oracle database for the UDDI registry” on page 3488
   d. To transfer your UDDI data, use the capabilities of the database products to export the data from the old database, and import it into the new one.
   e. Create the new data source. See “Creating a data source for the UDDI registry” on page 3489.
   f. Restart the UDDI registry application.
   g. Check that you can access your UDDI data, for example use the UDDI registry installation verification test, then delete the old database.

Creating a DB2 distributed database for the UDDI registry:

Complete this task if you want to use DB2 on the Windows, Linux, or UNIX operating systems as the database store for your UDDI registry data.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

DataBaseName
The name of the UDDI registry database. A suggested value is UDDI30. The UDDI information uses the suggested name of UDDI30, so if you use a different name, remember to substitute it when you see UDDI30 in the UDDI information.

DB2UserID
A DB2 user ID with administrative privileges.

DB2Password
The password for the DB2 user ID.

BufferPoolName
The name of a buffer pool for the UDDI registry database to use. A suggested value is uddibp, but you can use any name because the buffer pool is created as part of this task.

TableSpaceName
The name of a table space. A suggested value is uddits, but you can use any name.

TempTableSpaceName
The name of a temporary table space. A suggested value is udditstemp, but you can use any name because the temporary table space is created as part of this task.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Change the directory to *app_server_root*/UDDIReg/databaseScripts.
2. Start the DB2 Command Line Processor. Enter the following command at the command prompt:
   ```
   db2
   ```
3. Set up the DB2 environment variables. Enter the following command:
   ```
   set DB2CODEPAGE=1208
   ```
4. Create the DB2 database. Enter the following command:
   ```
   create database DataBaseName using codeset UTF-8 territory en
   ```
5. Configure the DB2 database. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   update db cfg for DataBaseName using applheapsz 2048
   ```
   c.
   ```
   update db cfg for DataBaseName using logfilsiz 8192
   ```
   d.
   ```
   connect reset
   ```

e.
```
terminate
```
f.
```
force application all
```
g.
```
terminate
```
h.
```
stop
```
i.
```
start
```

6. Restart the DB2 Command Line Processor. For all operating systems except Windows, enter the following command at the command prompt:
```
db2
```
7. Create further database structures. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   create regular tablespace uddits pagesize 32K managed by system using
   ('TableSpaceName') extentsize 64 prefetchsize 32 bufferpool BufferPoolName
   ```
   c.
   ```
   create system temporary tablespace TempTableSpacename pagesize 32K managed by
   system using ('TempTableSpacename') extentsize 32 overhead 14.06
   prefetchsize 32 transferrate 0.33 bufferpool BufferPoolName
   ```
8. Define the database structures that are needed to store the UDDI data.

   Exit the DB2 Command Line Processor and enter the following commands exactly as shown. Note that one step uses `-vf` rather than `-tvf`.
   a.
   ```
   db2 -tvf uddi30crt_10_prereq_db2.sql
   ```
   b.
   ```
   db2 -tvf uddi30crt_20_tables_generic.sql
   ```
   c.
   ```
   db2 -tvf uddi30crt_25_tables_db2udb.sql
   ```
   d.
   ```
   db2 -tvf uddi30crt_30_constraints_generic.sql
   ```
   e.
   ```
   db2 -tvf uddi30crt_35_constraints_db2udb.sql
   ```
   f.
   ```
   db2 -tvf uddi30crt_40_views_generic.sql
   ```
   g.
   ```
   db2 -tvf uddi30crt_45_views_db2udb.sql
   ```
   h.
   ```
   db2 -vf uddi30crt_50_triggers_db2udb.sql
   ```
   i.
   ```
   db2 -tvf uddi30crt_60_insert_initial_static_data.sql
   ```
9. Optional: To use the database as a default UDDI node, enter the following command:
   ```
   db2 -tvf uddi30crt_70_insert_default_database_indicator.sql
   ```
10. Issue the following commands:
    ```
    connect reset
    terminate
    ```
11. Issue the following commands:
    ```
    connect reset
    terminate
    ```

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating a DB2 for i database for the UDDI registry:

Complete this task if you want to use DB2 for i as the database store for your UDDI registry data.

Before you begin

The default names of the UDDI registry schema in the SQL scripts listed in the following topic are IBMUDI30 and IBMUDS30. These names are the recommended values and are assumed throughout the UDDI information. To use different names, modify the SQL files listed, then substitute the new names when IBMUDI30 and IBMUDS30 are used in the information center.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Use IBM Navigator for i to run SQL scripts.
   a. Open IBM Navigator for i.
   b. Expand **My Connections** > **iSeriesName** > **Databases**.
   c. Select **iSeriesName**.
   d. Right-click **Run SQL Scripts...**.

   A **Run SQL Scripts** window opens.
2. Open the IBM i DB2 SQL files.
   a. Map a network drive to the root directory of your IBM i server integrated file system.
   b. In Windows Explorer, expand the `WAS_HOME/UDDIReg/databaseScripts` directory.
   c. Open the following SQL files with a text editor (for example, Windows Notepad):
      - uddi30crt_10_prereq_db2_iSeries.sql
      - uddi30crt_20_tables_generic_iSeries.sql
      - uddi30crt_25_tables_db2udb_iSeries.sql
      - uddi30crt_30_constraints_generic_iSeries.sql
      - uddi30crt_35_constraints_db2udb_iSeries.sql
      - uddi30crt_40_views_generic_iSeries.sql
      - uddi30crt_45_views_db2udb_iSeries.sql
      - uddi30crt_50_triggers_db2udb_iSeries.sql
      - uddi30crt_60_insert_initial_static_data_iSeries.sql
3. Copy the text to the **Run SQL Scripts** window.
   a. In the text editor of the file `uddi30crt_10_prereq_db2_iSeries.sql`, click **Edit** > **Select All**.
   b. Click **Edit** > **Copy**.
   c. In the **Run SQL Scripts** window, click **Edit** > **Paste**.
   d. Click **Run** > **All**.
   e. After the script completes running, select all the SQL text and delete it from the **Run SQL Scripts** window.
   f. Repeat the previous steps for all the SQL scripts listed in step 2.
4. Optional: If you want to use the database as a default UDDI node, complete the following steps:
   a. Open `uddi30crt_70_insert_default_database_indicator.sql` as described in step 2.
   b. Copy and run `uddi30crt_70_insert_default_database_indicator.sql` as described in step 3.

What to do next

Continue to set up and deploy your UDDI registry node.

Creating an Apache Derby database for the UDDI registry:

Complete this task to use an Apache Derby database as the database store for your UDDI registry. You can use an embedded or network Apache Derby database, and the database store can be local or remote.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

arg1 The path of the SQL files. On a standard installation, the path is *app_server_root*/UDDIReg/databasescripts.

arg2 The path to the location where you want to install the Apache Derby database.

For example, *profile_root*/databases/com.ibm.uddi.

arg3 The name of the Apache Derby database. A recommended value is UDDI30, and this name is assumed throughout the UDDI information. If you use another name, substitute that name when UDDI30 is used in the UDDI information.

arg4 An optional argument. Either use the value DEFAULT, or omit this argument. Specify DEFAULT to use the database as a default UDDI node. This argument is case sensitive.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Start a Qshell session. Enter the STRQSH command from the IBM i command line.
2. Create a UDDI Apache Derby database by using UDDIDerbyCreate.jar. Run the following Java -jar command from the *app_server_root*/UDDIReg/databaseScripts directory.

```
java -Djava.ext.dirs=app_server_root/derby/lib:app_server_root/java/jre/lib/ext -jar UDDIDerbyCreate.jar
arg1 arg2 arg3 arg4
```

   If the Apache Derby database already exists, you are asked if you want to re-create it. If you choose to re-create the database, the existing database is deleted and a new one is created in its place. If you choose not to re-create the database, the command exits and a new database is not created.

   Note: If the application server has already accessed the existing Apache Derby database, the uddiDeploy.jacl script cannot re-create the database. Use the uddiRemove.jacl script to remove the database, as described in the topic about removing a UDDI registry node, restart the server, and run the uddiDeploy.jacl script again.
3. If you are using a remote database, which requires network Apache Derby, or if you want to use network Apache Derby for other reasons, for example, to use Apache Derby with a cluster, configure the Apache Derby Network Server framework. For details, see the section about managing the Derby Network Server in the Derby Server and Administration Guide.

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating an Oracle database for the UDDI registry:

Complete this task if you want to use Oracle as the database store for your UDDI registry data.

Before you begin

This task creates three new schemas: ibmuddi, ibmudi30 and ibmuds30. You cannot complete this task if schemas with these names exist already.

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

OracleUserID
The Oracle user ID to use to create the database.

OraclePassword
The password for the Oracle user ID.

The Oracle database must be a remote database; you cannot create a local database. Refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

The supported versions of Oracle are Version 9i and Version 10g. Each version has the following restrictions:

Table 329. UDDI restrictions and Oracle versions. The table lists the Version 9i and Version 10g restrictions for different UDDI parameters.

	Version 9i restrictions		**Version 10g restrictions**	
discoveryURL (Business)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
accessPoint (bindingTemplate)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
instanceParms (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 8192 characters		
overviewURL (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 4096 characters		
Digital Signature	maximum 4000 bytes			

Procedure

1. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_10_prereq_oracle.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_20_tables_generic.sql
```
2. Run one of the following commands, depending on the version of Oracle.
   - For Oracle Version 9i:

```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle_pre10g.sql
```

   - For Oracle Version 10g and later:

```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle.sql
```

3. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_30_constraints_generic.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_35_constraints_oracle.sql
```
   c.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_40_views_generic.sql
```
   d.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_45_views_oracle.sql
```
   e.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_50_triggers_oracle.sql
```
   f.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_60_insert_initial_static_data.sql
```
4. Optional: To use the database as a default UDDI node, run the following command:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_70_insert_default_database_indicator.sql
```

What to do next

Continue with setting up and deploying your UDDI registry node.

Removing a UDDI registry node

You can remove the UDDI registry application, delete the UDDI registry database, move a UDDI registry to another server or profile, or remove a UDDI registry node completely.

About this task

A UDDI registry node consists of the following elements:

- An enterprise application.
- A store of data that is referred to as the UDDI registry database and that uses a relational database management system.
- A way to connect the application to the data, that is, a data source and related elements.

All the data that relates to UDDI is stored in the UDDI database and therefore that data is separate from the UDDI application. Therefore, there are several options when you remove a UDDI registry node:

- You can remove a UDDI registry node from the application server without deleting the database. You delete only the UDDI application and any associated resources, such as the data source, and J2EE Connector Architecture (J2C) authentication data if it is used. You might do this for the following reasons:
  - You no longer want a UDDI facility on a particular application server. You can then move the UDDI registry node to a different application server.
  - You want to reinstall the application, for example to apply service changes or because the application has been corrupted.
- You can delete the UDDI registry database. If you do this, all UDDI data for that UDDI registry is lost. You might do this for the following reasons:
  - You want to use a different database product as the persistence store for the UDDI data.
  - You want to delete all the UDDI registry data and publish fresh data, for example after you complete a test cycle.
  - You want to initialize the UDDI registry node with new UDDI property settings, for example, to move from a default UDDI node to a customized UDDI node.
- You can move a UDDI registry to another server or profile.

  You might do this after you create a profile and you want to move the UDDI registry to the new profile.

- You can remove a UDDI registry node completely from an application server. You remove the UDDI registry application, the UDDI registry database, and the resources that are used to reference the UDDI registry database.

  You might do this to remove a UDDI registry that is used for testing after testing has finished.

To start a new UDDI registry node, you do not need to remove the UDDI application. Instead, you create a new replacement node by changing the data source that the UDDI application uses to access the new UDDI database.

Depending on what you want to do, complete one of the following steps.

Procedure

- To remove a UDDI registry node from the application server without deleting the database, complete the following:
  1. Start a Qshell session by entering the STRQSH command from the IBM i command line.
  2. Run the `uddiRemove.jacl` wsadmin script from the *app_server_root*/bin directory.

     The syntax of the command is as follows.

```
wsadmin [-profileName profile_name] -f uddiRemove.jacl
        node_name server_name [default]
```

     The attributes of the command are as follows:
     - `-profileName` *profile_name* is optional, and is the name of the profile in which the UDDI application is deployed. If you do not specify a profile, the default profile is used.
     - *node_name* and *server_name* are the names of the WebSphere Application Server node and the application server in which the UDDI application is deployed. These are the names that you specified when you deployed the UDDI application, for example when you ran the `uddiDeploy.jacl` script.
     - `default` is optional. Use this option only for the Apache Derby database and only if you ran the `uddiDeploy.jacl` script and used the default option to deploy the UDDI registry. This option removes the UDDI Apache Derby data source but does not remove the UDDI Apache Derby database.
  3. Optional: By default, output is displayed on screen. To direct the output to a log file, add the following to the end of the command, where *removeuddi*.log can be any name that you choose for the log file:

```
> removeuddi.log
```

     For example, to remove the UDDI application from server server1 that runs in node MyNode and send any messages to the file removeuddi.log:

```
wsadmin -profileName myProfile -f uddiRemove.jacl MyNode server1 > removeuddi.log
```

     Note: You can also remove the UDDI registry application by using the administrative console in the usual way, by selecting the application in the **Enterprise Applications** view and clicking **Uninstall**.
- To delete a UDDI registry database, complete the following steps. Remember that all UDDI data in the UDDI registry is deleted.
  1. Stop the server that hosts the UDDI registry application.
  2. Delete the database.
     - For DB2 for i, use either Navigator for i or a 5250 session to delete the IBMUDI30 and IBMUDS30 schemas.
     - For Oracle, delete the IBMUDDI, IBMUDI30 and IBMUDS30 schemas.
     - For Apache Derby, delete the directory tree that contains the UDDI database. By default, this directory tree is in the *profile_root*/databases/com.ibm.uddi/UDDI30 directory.
- To move a UDDI registry node to another server or profile, complete the following steps:

1. Ensure that the UDDI registry database remains accessible after the move. You might need to copy the database to a suitable new location. For example, if the database is remote, the new server must be able to access it. Also, the database might be deleted after the move. This situation occurs if you move the UDDI registry to a new profile and then delete the old profile, because any databases that were stored in the profile are also deleted. An example of such a database is an Apache Derby database that is created as part of creating default UDDI node.
2. Remove the UDDI registry application. See the step to remove a UDDI registry node from the application server.
3. Optional: Delete the data source and related objects.

   For the Apache Derby database, if you ran the `uddiRemove.jacl` script and used the default option to remove the UDDI registry application, the data source and related objects are deleted already and you do not need to complete this step. In all other situations, delete the following objects:
   - The UDDI data source that references the UDDI registry database, that is, the data source that was created when you set up the UDDI registry.
   - Any UDDI JDBC provider that was created if you did not reuse an existing JDBC provider.
   - Any J2C authentication data entry.
4. In the new server, if appropriate, create a J2C authentication data entry, and create a JDBC provider and a data source to reference the existing database. See the relevant steps in Setting up a customized UDDI node.
5. Deploy the UDDI registry application. See Deploying the UDDI registry application. If you use the supplied script, do not use the default option even if you used this option previously to set up a default UDDI node. Do not use the default option because an error might occur during deployment, or, in some circumstances, existing UDDI data might be overwritten.

   Note: The UDDI node name does not change. If the UDDI node name includes the node name and server name of the original server, after the move there is a mismatch between the UDDI node name, and the node name and server name of the new server. However, this mismatch does not affect the UDDI registry node function.
6. Check that the UDDI data can be accessed. If you are using a copy of the original UDDI registry database, you can now delete the original database. See the step to delete a UDDI registry database.

- To remove a UDDI registry node completely, complete the following steps:
  1. Remove the UDDI registry application. See the step to remove a UDDI registry node from the application server.
  2. Delete the UDDI registry database. See the step to delete a UDDI registry database.
  3. Optional: Delete the data source and related objects.

     For the Apache Derby database, if you ran the `uddiRemove.jacl` script and used the default option to remove the UDDI registry application, the data source and related objects are deleted already and you do not need to complete this step. In all other situations, delete the following objects:
     - The UDDI data source that references the UDDI registry database, that is, the data source that was created when you set up the UDDI registry.
     - Any UDDI JDBC provider that was created if you did not reuse an existing JDBC provider.
     - Any J2C authentication data entry.

What to do next

If you removed a UDDI registry node from the application server without deleting the database, you might want to reinstall the UDDI registry application.

Reinstalling the UDDI registry application

You can remove and reinstall an existing UDDI registry application to change the UDDI application code but continue to provide UDDI services with the existing UDDI database.

About this task

A UDDI registry node consists of the following elements:

- A Java EE application.
- A store of data that is referred to as the UDDI database. The UDDI database uses a relational database management system.
- A way to connect the application to the data (a data source and related elements).

All the data that relates to UDDI is stored in the UDDI database and therefore that data exists, regardless of the UDDI application. Therefore, you can remove a UDDI registry node from the application server without deleting the database, then reinstall the UDDI registry application. You might do this if an application is corrupted, or to apply service changes.

Procedure

1. Note any changes that you made to the installed UDDI application that you want to keep, for example changes to security role mappings, changes to the deployment descriptor (web.xml) in the v3soap.war, v3gui.war, v3soap.war, or soap.war files, or customization of the UDDI user interface (GUI). All such changes are lost during the reinstallation process, so you must reapply changes that you want to keep later.
2. Remove the existing UDDI application and reinstall it by running the uddiDeploy.jacl wsadmin script from the *app_server_root*/bin directory. Do not use the default option even if you used this option previously to set up a default UDDI node. If you use the default option, an error might occur during deployment, or, in some circumstances, existing UDDI data might be overwritten.

   Enter the following command at a command prompt:

```
wsadmin [-conntype none] [-profileName profile_name] -f uddiDeploy.jacl
       node_name server_name
```

   where:
   - `-conntype none` is optional, and is needed only if the application server is not running.
   - `-profileName` *profile_name* is optional, and is the name of the profile in which the UDDI application is deployed. If you do not specify a profile, the default profile is used.
   - *node_name* and *server_name* are the names of the WebSphere Application server node and the application server in which the UDDI application is deployed. These are the names that you specified when you ran the uddiDeploy.jacl script to install the UDDI application.

   Note: This procedure does not change the existing JDBC provider, data source and any J2EE Connector Architecture (J2C) authentication data entry. Your existing UDDI registry data, including UDDI entities, property settings, and policy settings, are also unaffected.
3. Optional: To direct the output to a log file, add the following option to the end of the command, where *log_name*.log can be any name that you choose for the log file:

```
> log_name.log
```

4. If you noted any changes in step 1, reapply them now.
5. For the reapplied changes to take effect, start or restart the application server.

Creating a DB2 distributed database for the UDDI registry

Complete this task if you want to use DB2 on the Windows, Linux, or UNIX operating systems as the database store for your UDDI registry data.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

DataBaseName
The name of the UDDI registry database. A suggested value is UDDI30. The UDDI information uses the suggested name of UDDI30, so if you use a different name, remember to substitute it when you see UDDI30 in the UDDI information.

DB2UserID
A DB2 user ID with administrative privileges.

DB2Password
The password for the DB2 user ID.

BufferPoolName
The name of a buffer pool for the UDDI registry database to use. A suggested value is uddibp, but you can use any name because the buffer pool is created as part of this task.

TableSpaceName
The name of a table space. A suggested value is uddits, but you can use any name.

TempTableSpaceName
The name of a temporary table space. A suggested value is udditstemp, but you can use any name because the temporary table space is created as part of this task.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Change the directory to *app_server_root*/UDDIReg/databaseScripts.
2. Start the DB2 Command Line Processor. Enter the following command at the command prompt:
   ```
   db2
   ```
3. Set up the DB2 environment variables. Enter the following command:
   ```
   set DB2CODEPAGE=1208
   ```
4. Create the DB2 database. Enter the following command:
   ```
   create database DataBaseName using codeset UTF-8 territory en
   ```
5. Configure the DB2 database. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   update db cfg for DataBaseName using applheapsz 2048
   ```
   c.
   ```
   update db cfg for DataBaseName using logfilsiz 8192
   ```
   d.
   ```
   connect reset
   ```
   e.
   ```
   terminate
   ```
   f.
   ```
   force application all
   ```
   g.
   ```
   terminate
   ```

h.
```
stop
```
i.
```
start
```
6. Restart the DB2 Command Line Processor. For all operating systems except Windows, enter the following command at the command prompt:
```
db2
```
7. Create further database structures. Enter the following commands:
   a.
   ```
   connect to DataBaseName user DB2UserID using DB2Password
   ```
   b.
   ```
   create regular tablespace uddits pagesize 32K managed by system using
   ('TableSpaceName') extentsize 64 prefetchsize 32 bufferpool BufferPoolName
   ```
   c.
   ```
   create system temporary tablespace TempTableSpacename pagesize 32K managed by
   system using ('TempTableSpacename') extentsize 32 overhead 14.06
   prefetchsize 32 transferrate 0.33 bufferpool BufferPoolName
   ```
8. Define the database structures that are needed to store the UDDI data.

   Exit the DB2 Command Line Processor and enter the following commands exactly as shown. Note that one step uses `-vf` rather than `-tvf`.
   a.
   ```
   db2 -tvf uddi30crt_10_prereq_db2.sql
   ```
   b.
   ```
   db2 -tvf uddi30crt_20_tables_generic.sql
   ```
   c.
   ```
   db2 -tvf uddi30crt_25_tables_db2udb.sql
   ```
   d.
   ```
   db2 -tvf uddi30crt_30_constraints_generic.sql
   ```
   e.
   ```
   db2 -tvf uddi30crt_35_constraints_db2udb.sql
   ```
   f.
   ```
   db2 -tvf uddi30crt_40_views_generic.sql
   ```
   g.
   ```
   db2 -tvf uddi30crt_45_views_db2udb.sql
   ```
   h.
   ```
   db2 -vf uddi30crt_50_triggers_db2udb.sql
   ```
   i.
   ```
   db2 -tvf uddi30crt_60_insert_initial_static_data.sql
   ```
9. Optional: To use the database as a default UDDI node, enter the following command:
```
db2 -tvf uddi30crt_70_insert_default_database_indicator.sql
```
10. Issue the following commands:
```
connect reset
terminate
```
11. Issue the following commands:
```
connect reset
terminate
```

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating a DB2 for i database for the UDDI registry

Complete this task if you want to use DB2 for i as the database store for your UDDI registry data.

Before you begin

The default names of the UDDI registry schema in the SQL scripts listed in the following topic are IBMUDI30 and IBMUDS30. These names are the recommended values and are assumed throughout the

UDDI information. To use different names, modify the SQL files listed, then substitute the new names when IBMUDI30 and IBMUDS30 are used in the information center.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Use IBM Navigator for i to run SQL scripts.
   a. Open IBM Navigator for i.
   b. Expand **My Connections** > **iSeriesName** > **Databases**.
   c. Select **iSeriesName**.
   d. Right-click **Run SQL Scripts...**.

   A **Run SQL Scripts** window opens.
2. Open the IBM i DB2 SQL files.
   a. Map a network drive to the root directory of your IBM i server integrated file system.
   b. In Windows Explorer, expand the `WAS_HOME/UDDIReg/databaseScripts` directory.
   c. Open the following SQL files with a text editor (for example, Windows Notepad):
      - uddi30crt_10_prereq_db2_iSeries.sql
      - uddi30crt_20_tables_generic_iSeries.sql
      - uddi30crt_25_tables_db2udb_iSeries.sql
      - uddi30crt_30_constraints_generic_iSeries.sql
      - uddi30crt_35_constraints_db2udb_iSeries.sql
      - uddi30crt_40_views_generic_iSeries.sql
      - uddi30crt_45_views_db2udb_iSeries.sql
      - uddi30crt_50_triggers_db2udb_iSeries.sql
      - uddi30crt_60_insert_initial_static_data_iSeries.sql
3. Copy the text to the **Run SQL Scripts** window.
   a. In the text editor of the file `uddi30crt_10_prereq_db2_iSeries.sql`, click **Edit** > **Select All**.
   b. Click **Edit** > **Copy**.
   c. In the **Run SQL Scripts** window, click **Edit** > **Paste**.
   d. Click **Run** > **All**.
   e. After the script completes running, select all the SQL text and delete it from the **Run SQL Scripts** window.
   f. Repeat the previous steps for all the SQL scripts listed in step 2.
4. Optional: If you want to use the database as a default UDDI node, complete the following steps:
   a. Open `uddi30crt_70_insert_default_database_indicator.sql` as described in step 2.
   b. Copy and run `uddi30crt_70_insert_default_database_indicator.sql` as described in step 3.

What to do next

Continue to set up and deploy your UDDI registry node.

Creating an Apache Derby database for the UDDI registry

Complete this task to use an Apache Derby database as the database store for your UDDI registry. You can use an embedded or network Apache Derby database, and the database store can be local or remote.

Before you begin

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

arg1 The path of the SQL files. On a standard installation, the path is *app_server_root*/UDDIReg/databasescripts.

arg2 The path to the location where you want to install the Apache Derby database.

For example, *profile_root*/databases/com.ibm.uddi.

arg3 The name of the Apache Derby database. A recommended value is UDDI30, and this name is assumed throughout the UDDI information. If you use another name, substitute that name when UDDI30 is used in the UDDI information.

arg4 An optional argument. Either use the value DEFAULT, or omit this argument. Specify DEFAULT to use the database as a default UDDI node. This argument is case sensitive.

If you want to create a remote database, refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

Procedure

1. Start a Qshell session. Enter the STRQSH command from the IBM i command line.
2. Create a UDDI Apache Derby database by using UDDIDerbyCreate.jar. Run the following Java -jar command from the *app_server_root*/UDDIReg/databaseScripts directory.

```
java -Djava.ext.dirs=app_server_root/derby/lib:app_server_root/java/jre/lib/ext -jar UDDIDerbyCreate.jar
arg1 arg2 arg3 arg4
```

   If the Apache Derby database already exists, you are asked if you want to re-create it. If you choose to re-create the database, the existing database is deleted and a new one is created in its place. If you choose not to re-create the database, the command exits and a new database is not created.

   Note: If the application server has already accessed the existing Apache Derby database, the uddiDeploy.jacl script cannot re-create the database. Use the uddiRemove.jacl script to remove the database, as described in the topic about removing a UDDI registry node, restart the server, and run the uddiDeploy.jacl script again.
3. If you are using a remote database, which requires network Apache Derby, or if you want to use network Apache Derby for other reasons, for example, to use Apache Derby with a cluster, configure the Apache Derby Network Server framework. For details, see the section about managing the Derby Network Server in the Derby Server and Administration Guide.

What to do next

Continue with setting up and deploying your UDDI registry node.

Creating an Oracle database for the UDDI registry

Complete this task if you want to use Oracle as the database store for your UDDI registry data.

Before you begin

This task creates three new schemas: ibmuddi, ibmudi30 and ibmuds30. You cannot complete this task if schemas with these names exist already.

The following steps use a number of variables. Before you start, decide appropriate values to use for these variables. The variables, and suggested values, are:

OracleUserID
The Oracle user ID to use to create the database.

OraclePassword
The password for the Oracle user ID.

The Oracle database must be a remote database; you cannot create a local database. Refer first to the database product documentation about the relevant capabilities of the product.

About this task

You complete this task only once for each UDDI registry, as part of setting up and deploying a UDDI registry.

The supported versions of Oracle are Version 9i and Version 10g. Each version has the following restrictions:

Table 330. UDDI restrictions and Oracle versions. The table lists the Version 9i and Version 10g restrictions for different UDDI parameters.

	Version 9i restrictions		**Version 10g restrictions**	
discoveryURL (Business)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
accessPoint (bindingTemplate)	maximum 4000 bytes	UDDI specification 4096 characters	maximum 4000 bytes	UDDI specification 4096 characters
instanceParms (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 8192 characters		
overviewURL (tModelInstanceInfo)	maximum 4000 bytes	UDDI specification 4096 characters		
Digital Signature	maximum 4000 bytes			

Procedure

1. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_10_prereq_oracle.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_20_tables_generic.sql
```
2. Run one of the following commands, depending on the version of Oracle.
   - For Oracle Version 9i:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle_pre10g.sql
```
   - For Oracle Version 10g and later:
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_25_tables_oracle.sql
```
3. Run the following commands:
   a.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_30_constraints_generic.sql
```
   b.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_35_constraints_oracle.sql
```
   c.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_40_views_generic.sql
```
   d.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_45_views_oracle.sql
```
   e.
```
sqlplus OracleUserID/OraclePassword @ uddi30crt_50_triggers_oracle.sql
```

f.

```
sqlplus OracleUserID/OraclePassword @ uddi30crt_60_insert_initial_static_data.sql
```

4. Optional: To use the database as a default UDDI node, run the following command:

```
sqlplus OracleUserID/OraclePassword @ uddi30crt_70_insert_default_database_indicator.sql
```

What to do next

Continue with setting up and deploying your UDDI registry node.

Applying an upgrade to the UDDI registry

You can apply an interim fix, a fix pack, or a refresh pack to the UDDI registry.

Procedure

1. Apply the WebSphere Application Server interim fix, fix pack, or refresh pack to your application server, or servers, by using the WebSphere Application Server Update Installer. Repeat this process for each server that you want to apply the UDDI upgrade to. If you have not deployed a UDDI registry yet, no further action is required, because updates to the UDDI registry take effect when you first deploy UDDI into any of your application server profiles.
2. If you already deployed a UDDI registry to one or more application server profiles, to apply the upgrade, redeploy the UDDI application, as described in Reinstalling the UDDI registry application. The existing UDDI application is removed and the updated application is deployed.

What to do next

Some upgrades might require additional steps. Refer to the readme file for the upgrade to check whether there are additional steps for this upgrade.

Configuring SOAP API and GUI services for the UDDI registry

For SOAP application programming interface (API) and graphical user interface (GUI) UDDI services, you can configure whether the service is secure. For Version 1 and Version 2 SOAP API services, you can also configure the default pool size.

Before you begin

The UDDI registry application must be installed.

About this task

For the Version 1 and Version 2 SOAP interface, you can configure the following properties:

- **defaultPoolSize**. The number of SOAP parsers with which to initialize the parser pool for the SOAP interface. You can set this property independently for the Publish (uddipublish) and Inquiry (uddi) APIs. For example, if you expect more inquiries than publish requests through the SOAP interface, you can set a larger pool size for the Inquiry API. The default initial size for both APIs is 10.
- Whether the API is secure, that is, accessed using HTTPS, or insecure, that is, accessed using HTTP. The default is to use HTTPS for the Publish API and HTTP for the Inquiry API.

For the Version 3 SOAP interface, you can specify whether Publish, Custody Transfer, Security and Inquiry APIs are secure, that is, accessed using HTTPS, or insecure, that is, accessed using HTTP. The default is to use HTTPS for the Publish, Custody Transfer, and Security APIs, and HTTP for the Inquiry API.

For the Version 3 GUI interface, you can specify whether the Publish and Inquiry services are secure, that is, accessed using HTTPS, or insecure, that is, accessed using HTTP. The default is to use HTTPS for the Publish service and HTTP for the Inquiry service.

Procedure

- Optional: To configure Version 1 and Version 2 SOAP API services, use the following steps.
  1. Edit the active deployment descriptor, web.xml, for the Version 1 and Version 2 SOAP module, soap.war. The web.xml file is in the following directory:

profile_root/config/cells/*cell_name*/applications/
UDDIRegistry.*node_name*.*server_name*.ear/deployments/
UDDIRegistry.*node_name*.*server_name*/soap.war/WEB-INF

  2. To modify the defaultPoolSize parameter for the Version 1 or Version 2 Publish API, modify the param-value element in the servlet with servlet-name = uddipublish.
  3. To modify the defaultPoolSize parameter for the Version 1 or Version 2 Inquiry API, modify the param-value element in the servlet with servlet-name = uddi.
  4. To set whether the Publish service is secure or insecure for the Version 1 or Version 2 Publish API, modify the user data constraint transport guarantee. Find the security-constraint element with id = UDDIPublishTransportConstraint. Set the user-data-constraint transport-guarantee for that element to CONFIDENTIAL, that is, the service is secure and is accessed using HTTPS, or NONE, that is, the service is insecure and is accessed using HTTP.
  5. Stop and restart the application server for the changes to take effect.
- Optional: For the Version 3 SOAP interface, specify whether the Publish, Custody Transfer, Security, and Inquiry API services are secure or insecure by using the following steps. The default is to use HTTPS for the Publish, Custody Transfer, and Security APIs, and HTTP for the Inquiry API.
  1. Edit the active deployment descriptor, web.xml, for the Version 3 SOAP module, v3soap.war. The web.xml file is in the following directory:

profile_root/config/cells/*cell_name*/applications/
UDDIRegistry.*node_name*.*server_name*.ear/deployments/
UDDIRegistry.*node_name*.*server_name*/v3soap.war/WEB-INF

     Each type of service is represented by a <security-constraint> element in the web.xml file. Each <security-constraint> element includes the <display-name> element and <user-data-constraint> <transport-guarantee> element.
  2. For each service that you want to modify, search the web.xml file for the relevant <display-name> value (see the following table) to locate the <security-constraint> element for that service. Set the <user-data-constraint> <transport-guarantee> for that element to CONFIDENTIAL, that is, the service is secure and is accessed using HTTPS, or NONE, that is, the service is insecure and is accessed using HTTP.

Table 331. Display name values in web.xml for UDDI services. The table lists the different types of UDDI service and the value of the <security-constraint> <display-name> element for each one.

Type of UDDI service	Value of <security-constraint> <display-name> element
Publish	AxisServlet Publish Resource Collection
Custody transfer	AxisServlet CustodyTransfer Resource Collection
Security	AxisServlet Security Resource Collection
Inquiry	AxisServlet Inquiry Resource Collection

  3. Stop and restart the application server for the changes to take effect.
- Optional: For the Version 3 GUI interface, specify whether the Publish and Inquiry services are secure or insecure by using the following steps. The default is to use HTTPS for the Publish service and HTTP for the Inquiry service.
  1. Edit the active deployment descriptor, web.xml, for the Version 3 GUI module, v3gui.war. The web.xml file is in the following directory:

profile_root/config/cells/*cell_name*/applications/
UDDIRegistry.*node_name*.*server_name*.ear/deployments/
UDDIRegistry.*node_name*.*server_name*/v3gui.war/WEB-INF

  2. For each service that you want to modify, search the web.xml file for the relevant <user-data-constraint> value (see the following table). Set the <user-data-constraint>

<transport-guarantee> for that element to CONFIDENTIAL, that is, the service is secure and is accessed using HTTPS, or NONE, that is, the service is insecure and is accessed using HTTP.

Table 332. User data constraint ids in web.xml for UDDI services. The table lists the different types of UDDI service and the value of the <user-data-constraint id> element for each one.

Type of UDDI service	Value of <user-data-constraint id> element
Publish	UDDIPublishTransportConstraint
Inquiry	UDDIInquireTransportConstraint

3. Stop and restart the application server for the changes to take effect.

Managing the UDDI registry

You can use the WebSphere Application Server administrative console or the Java Management Extensions (JMX) management interface to manage all the policies and properties of the UDDI registry.

About this task

You can use JMX to monitor and configure UDDI registries programmatically, and use the UDDI registry administrative interface.

To manage UDDI nodes programmatically, you can use the following interfaces and tools:

- UDDI registry administrative (JMX) interface

  The UDDI registry administrative (JMX) interface provides a Java API that you can use to manage runtime configuration settings to control UDDI registry runtime behavior, such as setting the maximum number of results that UDDI users can receive for inquiry requests, or creating publish limits for UDDI publishers. Sample client code is provided for you to build on.
- User-defined value sets

  The UDDI Version 3 registry provides tools that you can use to manage your own categorization value sets, including loading value set data into a UDDI registry node. If you add custom value sets, UDDI entities can be categorized more specifically when they are published, so that clients can find specific data more efficiently.
- UDDI Utility Tools

  The UDDI Utility Tools is a suite of functions and a Java API that you can use to promote Version 2 entities from one UDDI registry to another while retaining entity keys. This capability is particularly useful for publishing canonical tModels with a predefined key.

To manage UDDI registries by using the WebSphere Application Server administrative console, start from the UDDI link in the navigation pane of the administrative console. If administrative security is enabled, you must log in to the administrative console and supply a valid user ID and password, to use the UDDI management functions. You can complete the following operations:

Procedure

- View and manage the status of all UDDI nodes in a cell.
- Initialize UDDI nodes with required settings.
- Configure general properties that affect UDDI runtime behavior.
- Manage UDDI policy settings.
- Create, view and update UDDI publishers.
- Create, view and update publisher tiers that limit how many UDDI entries can be published.
- View and manage the status of value sets.

Backing up and restoring the UDDI registry database

If you want to protect the data in your UDDI registry database, you can back up and restore the database by using the facilities of the database product that your UDDI node is on.

Procedure

- To back up an Apache Derby UDDI registry database, use the following steps:
  1. Ensure that the UDDI application is stopped, and therefore is not accessing the Apache Derby database.
  2. Ensure that no other application is using the Apache Derby UDDI30 database.
  3. Copy the UDDI30 directory by using the file system that the directory is in.
- To restore an Apache Derby UDDI registry database, replace the UDDI30 file structure with the backup copy. Remember that any updates made after the backup was taken are lost.
- To back up a UDDI registry database that does not use Apache Derby, use the appropriate backup and restore tools for the database. To use these tools, refer to the documentation for the database product.

UDDI node collection

You can manage the UDDI nodes in this cell. Each UDDI node represents an individual UDDI registry application. A UDDI node is displayed in this list only if its underlying UDDI application is started. The status of the UDDI node can indicate whether the node is activated (available to accept API requests), deactivated (not allowing user requests), or not initialized. UDDI nodes that are not initialized require some properties to be set before they can be initialized and activated.

To view this administrative console page, click **UDDI** > **UDDI Nodes**.

UDDI Node ID:

Specifies the identifier for the UDDI node.

To manage an individual UDDI node, click on the ***UDDI _node_id*** to display the UDDI node settings page, where you can manage its general properties, initialize it if the status is set to `Initialization Pending`, and access pages for managing policies, UDDI publishers, tiers and value sets.

Description:

Specifies the description of the UDDI node.

UDDI Application Location:

Specifies the server in which the UDDI registry application is deployed and running.

Status:

Specifies the status of the UDDI node.

The UDDI node can have one of the following statuses:

- `Not initialized`
- `Initialization pending`
- `Initialization in progress`
- `Migration in progress`
- `Migration pending`
- `Value set creation in progress`
- `Value set creation pending`
- `Activated`

- `Deactivated`

If the status of a node is `Initialization pending`, you must initialize the node before you can activate it. If you attempt to initialize the node and it remains in a pending state, an error occurred during migration or initialization.

To activate UDDI nodes that are deactivated, select them using the corresponding check boxes and click **Activate**. Similarly, to deactivate UDDI nodes, select them and click **Deactivate**.

Note: If you restart the UDDI application, or the application server, the UDDI node is always reactivated, even if the node was previously deactivated.

UDDI node settings:

You can configure the general properties for a UDDI node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id***.

The UDDI node detail page displays general properties for the UDDI node. Depending on the status of the node, you can edit some properties. Use the links on this page to view or change additional properties, such as Value sets, Tiers and UDDI Publishers. Use the Policy Groups link on this page to view or change UDDI node policy.

Unless the UDDI node is installed as a default UDDI node, you must set certain general properties before you can initialize the UDDI node. These required properties are indicated by an asterisk (*) next to the relevant fields. You can set these property values as many times as you want before you initialize the UDDI node. However, after initialization, these properties become read-only for the lifetime of that UDDI node. Therefore, it is important to set the required properties correctly. You can set other general properties of the UDDI node both before and after initialization.

After you set the general properties to appropriate values, click **OK** to save your changes and exit the page, or **Apply** to save your changes and remain on the same page. At this point, the changes are stored.

If the status of the UDDI node is “Not initialized”, an **Initialize** option is displayed. To initialize the UDDI node, ensure that you save any changes to the general properties by clicking **Apply** or **OK**, then click **Initialize**. This operation might take some time to complete.

UDDI node ID:

Specifies the unique identifier for a UDDI node in a UDDI registry. The node ID must be a valid UDDI key. The value is also the domain key for the UDDI node.

Required	Yes
Data type	String
Default	uddi:*cell_name*:*node_name*:*server_name*:node_id

UDDI node description:

Specifies the description of this UDDI node.

Required	Yes
Data type	String
Default	WebSphere UDDI registry default node

Root key generator:

Specifies the root key space of the registry. For registries that can become affiliate registries, you might want to specify a root key space in a partition below the root key generator of the parent root registry, for example, `uddi:thisregistry.com:keygenerator`.

Required	Yes
Data type	String
Default	uddi:*cell_name*:*node_name*:*server_name*:keyspace_id:keygenerator

Prefix for generated discoveryURLs:

Specifies the URL prefix that is applied to generated discoveryURLs in businessEntity elements, so that the discoveryURLs can be returned on HTTP GET requests. This property applies to UDDI version 2 API requests only. Set this prefix to a valid URL for your configuration, and do not change it unless absolutely necessary.

The format is `http://hostname:port/uddisoap/`, where `uddisoap` is the context root of the UDDI version 2 SOAP servlet.

Although this field is not required, you set it so that the required and valid URL is generated in response to version 2 GET requests. After you set the prefix, do not change it unless it becomes invalid following a later configuration change. If you change the prefix, any discoveryURLs that were generated by using the earlier prefix no longer work.

Required	No
Data type	String
Default	http://localhost:9080/uddisoap

Host name for UDDI node services:

Specifies the host name root that the UDDI node uses to model API services in its own node business entity. This value must be the fully qualified domain name, or IP address, of the network host.

The UDDI node provides web services that implement each of the UDDI API sets that it supports. The host name is used to generate access point URLs in the bindingTemplate elements for each of the services. The access point URL is generated by prefixing the host name value with a protocol, such as http, and suffixing it with the corresponding host port number. The access point URL must resolve to a valid URL.

Data type	String
Default	localhost

Host HTTP port:

Specifies the port number that is used to access UDDI node services with HTTP. This port number must match the WebSphere Application Server port for HTTP requests.

Data type	Integer
Default	9080

Host HTTPS port:

Specifies the port number that is used to access UDDI node services with HTTPS. This port number must match the WebSphere Application Server port for HTTPS requests.

Data type	Integer
Default	9443

Maximum inquiry result set size:

Specifies the maximum size of the result set that the registry processes for an inquiry API request.

If the result set exceeds this value, an E_resultSetTooLarge error is returned. If you set this value too low, and users use imprecise search criteria, it is more likely that an E_resultSetTooLarge error is returned. If you set this value higher, result sets are larger, but response times might increase.

Data type	Integer
Default	500
Range	0 to 1024

Maximum inquiry response set size:

Specifies the maximum number of results that are returned in each response for inquiry API requests. Do not set this value higher than the value of **Maximum inquiry result set size**.

If the result set contains more results than this value, the response includes only a subset of those results. The user can retrieve the remaining results by using the listDescription feature as described in the UDDI specification. If you set this value too low, the user must make more requests to retrieve the remainder of the result set.

Data type	Integer
Default	500
Range	0 to 1024

Maximum search names:

Specifies the maximum number of names that can be supplied in an inquiry API request. If you set higher values, the UDDI node can process more complex requests, but complex requests can increase the response times of the UDDI node significantly. Therefore, to avoid increasing UDDI node response times, set this value to 8 or less.

Data type	Integer
Default	5
Range	1 to 64

Maximum search keys:

Specifies the maximum number of keys that can be supplied in an inquiry API request. If you set higher values, the UDDI node can process more complex requests, but complex requests can increase the response times of the UDDI node significantly. Therefore, to avoid increasing UDDI node response times, set this value to 5 or less.

This value limits the number of references that can be specified in categoryBag, identifierBag, tModelBag and discoveryURLs elements.

In exceptional cases, the UDDI node might reject complex requests with too many keys, even if the value of maxSearchKeys is not exceeded.

Data type Integer
Default 5
Range 1 to 64

Key space requests require digital signature:

Specifies whether tModel:keyGenerator requests must be digitally signed.

Data type Boolean (check box)
Default False (cleared)

Use tier limits:

Specifies whether an approval manager is used to check publication tier limits. If you set this value to `false`, an unlimited number of UDDI entities can be published.

Data type Boolean (check box)
Default True (selected)

Use authInfo credentials if provided:

Specifies whether authInfo contents in UDDI API requests are used to validate users when WebSphere Application Server administrative security is off. If you select this option, the UDDI node uses the authInfo element in the request. If you clear this option, the UDDI node uses the default user name.

Data type Boolean (check box)
Default True (selected)

Authentication token expiry period:

Specifies the period, in minutes, after which an authentication token is invalidated and a new authentication token is required.

Set this value high enough to allow the registry to operate successfully, but be aware that high values can increase the risk of illegal use of authentication tokens.

Data type Integer
Default 30
Range 1 to 10080 minutes (10080 minutes = 1 week)

Automatically register UDDI publishers:

Specifies whether UDDI publishers are automatically registered and assigned to the default tier. Automatically registered UDDI publishers are given default entitlements.

Data type Boolean (check box)
Default True (selected)

Default user name:

Specifies the user name that is used for publish operations when WebSphere Application Server administrative security is off and **Use authInfo credentials if provided** is set to false.

Data type	String
Default	UNAUTHENTICATED

Default language code:

Specifies, for UDDI version 1 and version 2 requests, the default language code to be used for the xml:lang element, when it is not otherwise specified.

Data type	String
Default	en

Value set collection:

You can view and configure the value sets that are installed in a UDDI node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Value Sets**.

Value sets in a UDDI node are either supported or not supported by policy. By default, new value sets are not supported. After you publish a value set tModel entity and load value set data, you can control whether other UDDI entities can reference this value set tModel entity by setting the Supported policy.

To enable support for one or more value sets, select the value sets by using the corresponding check boxes in the Select column, then click **Enable Support**. The Supported field for all the selected value sets is updated to a value of `true` to show the new status.

You might need to remove support for a value set before you remove the value set from the UDDI node. To remove support for a value set, select the corresponding check box, then click **Disable Support**. The corresponding Supported field is updated to a value of `false` to show the new status.

To view the attributes of a value set, click on the value set name in the list to display the Value set settings page.

Name:

Specifies the name of the tModel entity that represents this value set.

tModelkey:

Specifies the key for the tModel entity that represents this value set.

Supported:

Specifies whether this value set is supported by policy in this UDDI node, where `true` means supported, and `false` means not supported.

Value set settings:

You can view the attributes of a value set in a UDDI node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Value Sets** > ***value_set_name***.

This page shows the values of the keyedReference elements in the tModel entity that represents this value set. This page also shows the Supported status of the value set. All properties are read-only. To change the Supported status, use the Value sets collection page.

Unvalidatable:

Specifies whether this value set is categorized as unvalidatable, as described in the UDDI specification. The value set tModel publisher sets this value, to indicate whether the value set is available for use by publish requests.

Checked:

Specifies whether this value set is categorized as checked, as described in the UDDI specification. When this value is true, UDDI entities that reference this value set are validated to ensure that their values are present in this value set.

Cached:

Specifies whether this value set is cached in this UDDI node.

Externally cacheable:

Specifies whether this value set is externally cacheable.

Externally validated:

Specifies whether this value set is externally validated.

Supported:

Specifies whether this value set is supported by policy in this UDDI node.

Last cached:

Specifies the date when this value set was last cached in the UDDI node.

Tier collection:

You can view a list of the available tiers for the UDDI node. You can create new tiers, modify tiers, set the default tier, and delete tiers.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Tiers**.

To view or change the properties of a tier, click on the tier name in the list to display the UDDI Tier settings page.

To create a tier, click **New**. The UDDI Tier settings page is displayed, where you can set properties for the new tier.

One tier in the collection is marked as the default tier, indicated by the word (`default`) next to the tier name. The default tier is assigned to UDDI publishers that are registered automatically when automatic user registration is turned on. To set the default tier, select the tier by using the corresponding check box in the Select column, then click **Set default**.

To delete a tier, select the tier by using the corresponding check box in the Select column, then click **Delete**. You cannot delete a tier that is marked as the default tier, or that is currently assigned to a UDDI publisher.

Name:

Specifies the name of the tier.

Description:

Specifies the description of the tier.

UDDI Tier settings:

You can configure the general properties of a UDDI publisher tier.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Tiers** > ***tier_name***.

Name:

Specifies the name of the tier.

Required	Yes
Data type	String
Default	No default
Range	1 to 255

Description:

Specifies a description of the purpose or usage of this tier.

Data type	String
Default	No default
Range	0 to 255

Maximum properties:

The data for each maximum property field has the following properties:

Required	Yes
Data type	Integer
Default	No default
Range	0 to 2147483647

Maximum businesses:

Specifies the maximum number of businesses that UDDI publishers in this tier can publish.

Maximum services per business:

Specifies the maximum number of services that UDDI publishers in this tier can publish for each business.

Maximum bindings per service:

Specifies the maximum number of bindings that UDDI publishers in this tier can publish for each service.

Maximum tModels:

Specifies the maximum number of tModels that UDDI publishers in this tier can publish.

Maximum publisher assertions:

Specifies the maximum number of publisher assertions that UDDI publishers in this tier can add.

UDDI Publisher collection:

You can view the users that are currently registered as UDDI publishers. You can create a UDDI publisher, register a user as a UDDI publisher, assign a UDDI publisher to a tier, and delete a UDDI publisher.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **[Additional Properties] UDDI Publishers**.

To create a UDDI publisher, click **New**. The UDDI Publisher settings page is displayed, where you can set properties for the new UDDI publisher.

To register one or more existing WebSphere Application Server users as UDDI publishers, click **Create publishers**. The Create UDDI publishers page is displayed, where you can select users and modify their entitlements.

After users are registered as UDDI publishers, you can click on the user name to display the UDDI Publisher settings page and view or edit their entitlements.

You can assign multiple UDDI publishers to a tier without editing each UDDI publisher individually, by using the following steps:

1. Select the appropriate UDDI publishers by using the corresponding check boxes in the Select column.
2. From the tier list at the top of the collection table, select one of the tiers that is available on the UDDI node.
3. Click **Assign tier**.

To delete one or more UDDI publishers, select the UDDI publishers using the corresponding check boxes in the Select column, then click **Delete**.

User name:

Specifies the name of the UDDI publisher.

Tier:

Specifies the tier to which the UDDI publisher is assigned.

Create UDDI Publishers:

Use this page to register one or more existing WebSphere Application Server users as UDDI publishers.

To view this administrative console page, click **UDDI** > **UDDI Nodes >** *UDDI_node_id* **> [Additional Properties] UDDI Publishers** > **Create publishers**.

To register as UDDI publishers one or more users that are known to the application server, complete the following steps:

1. Enter a string to search for the required users. To find all users, use the * character.
2. Optionally, enter a number in the limit field to restrict the number of returned results.
3. Click **Search** to display a list of users that match the string.
4. Select the users that you want to register from the **Available** list and use the arrows to move them into the **Selected** list.
5. Use the entitlements listed under **General Properties** to give the UDDI publishers permission to undertake specific actions. Set the entitlements by selecting the check box next to each entitlement.

6. Select a tier for the users from the **Tier** list.
7. Click **OK** to register the users as UDDI publishers with the specified entitlements and tier.

Note: If you are using a Lightweight Directory Access Protocol (LDAP) user registry, the format of the name that is given to each UDDI Publisher is defined by the **User ID map** value in the LDAP advanced settings. To view the LDAP advanced settings, click **Security** > **Global security**, under **User account repository** select **Standalone LDAP registry** and click **Configure**, then under **Additional Properties** click **Advanced Lightweight Directory Access Protocol (LDAP) user registry settings**.

After a user has been registered as a UDDI publisher, you can edit their entitlements by clicking the user name.

Allowed to publish keyGenerator with derived key:

Specifies whether the UDDI publisher has permission to publish tModel: keyGenerator with a derived key.

The tModel:keyGenerator is a request for key space. An example of a legal derived key is uddi:tempuri.com:fish:buyingService where the key is based on the derivedKey "uddi:tempuri.com:fish". the string 'buyingService' is the key's key specific string (KSS).

Allowed to publish keyGenerator with domain keys:

Specifies whether the UDDI publisher has permission to publish tModel: keyGenerator with a domain key.

Allowed to publish keyGenerator:

Specifies whether the UDDI publisher has permission to publish tModel: keyGenerator.

If false, UDDI publishers cannot publish keyGenerators of any kind. In this situation all the entitlement settings are disregarded, regardless of how they are set.

Allowed to publish with UUID key:

Specifies whether the UDDI publisher has permission to publish elements with a UUID key.

Allowed to publish keyGenerator with UUID keys:

Specifies whether the UDDI publisher has permission to publish tModel: keyGenerator with a UUID key.

Tier:

Specifies the tier to which the UDDI publisher is assigned.

UDDI Publisher settings:

You can view and edit the entitlements and publication limits tier for a UDDI publisher, or create a new UDDI publisher.

To view this administrative console page, use one of the following options:

- To view and edit the properties of an existing UDDI publisher, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **UDDI Publishers** > ***user_name***.
- To create a new UDDI publisher, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **UDDI Publishers** > **New**.

User name:

Specifies the name of the UDDI publisher.

For a new UDDI publisher, enter the name of a user that is known to the application server. For an existing publisher, you cannot change the user name.

Allowed to publish keyGenerator with derived key:

Specifies whether the UDDI publisher has permission to publish a tModel:keyGenerator request with a derived key.

The tModel:keyGenerator request is a request for key space. An example of a legal derived key is uddi:tempuri.com:fish:buyingService, where the key is based on the derivedKey key uddi:tempuri.com:fish and the string buyingService is the key-specific string (KSS) for that key.

Data type Boolean
Default True (selected)

Allowed to publish keyGenerator with domain keys:

Specifies whether the UDDI publisher has permission to publish a tModel:keyGenerator request with a domain key.

Data type Boolean
Default True (selected)

Allowed to publish keyGenerator:

Specifies whether the UDDI publisher has permission to publish a tModel:keyGenerator request.

If you set this value to false, the UDDI publisher cannot publish keyGenerator requests of any kind. In this situation, the following settings are ignored, regardless of how they are set:

- **Allowed to publish keyGenerator with derived key**
- **Allowed to publish keyGenerator with domain keys**
- **Allowed to publish with UUID key**
- **Allowed to publish keyGenerator with UUID keys**

Data type Boolean
Default True (selected)

Allowed to publish with UUID key:

Specifies whether the UDDI publisher has permission to publish elements with a UUID key.

Data type Boolean
Default False (cleared)

Allowed to publish keyGenerator with UUID keys:

Specifies whether the UDDI publisher has permission to publish a tModel:keyGenerator request with a UUID key.

Data type Boolean
Default False (cleared)

Tier:

Specifies the tier to which the UDDI publisher is assigned.

Policy groups:

You can access the detailed settings information for every policy group that you can configure for a UDDI registry node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id***.

The page shows a list of the policy groups that can be acted upon. Click a specific group to open the page for the required group.

UDDI keying policy settings:

You can view or edit the UDDI keying settings for a UDDI registry.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Keying policies**.

Registry key generation:

Specifies whether publishers are allowed to publish key generator tModel entities. You can manage how publishers are allowed to publish key generator tModel entities by configuring the entitlements in the UDDI Publishers page.

Data type	Boolean
Default	True (selected)

Registry support of UUID keys:

Specifies whether publisher supplied uuidKeys keys are allowed in publish requests. You can manage how publishers are allowed to use uuidKeys keys by configuring the entitlements in the UDDI Publishers page.

Data type	Boolean
Default	False (cleared)

UDDI node API policy settings:

For UDDI Version 3, you can view or edit the API settings for a UDDI registry node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **API policies**.

Note: This information applies only to UDDI Version 3; you cannot change the API settings for Versions 1 and 2. In Versions 1 and 2, authentication tokens are required for publish requests, but not for inquiry requests. Versions 1 and 2 do not have custody transfer requests.

Authorization for inquiry:

Specifies whether authorization that uses the authInfo element is required for inquiry API requests. This setting is relevant only if the V3SOAP_Inquiry_User_Role role is set to Everyone and WebSphere Application Server administrative security is on.

If WebSphere Application Server administrative security is off, this setting is ignored. If WebSphere Application Server administrative security is on, but the V3SOAP_Inquiry_User_Role role is not set to Everyone, this setting is ignored.

If this option is selected, an authorization token is required to complete the request. If this option is cleared, an authorization token is not required. If this option is cleared and an authorization token is supplied, the token is ignored and the request is processed as if the default user that is defined in the UDDI node settings made the request.

Typically, UDDI registries are configured to not require authorization for inquiry API requests.

Data type Boolean
Default False (cleared)

Authorization for publish:

Specifies whether authorization that uses the authInfo element is required for publish API requests. This setting is relevant only if the V3SOAP_Publish_User_Role role is set to Everyone and WebSphere Application Server administrative security is on.

If WebSphere Application Server administrative security is off, this setting is ignored. If WebSphere Application Server administrative security is on, but the V3SOAP_Publish_User_Role role is not set to Everyone, this setting is ignored.

If this option is selected, an authorization token is required to complete the request. If this option is cleared, an authorization token is not required. If this option is cleared and an authorization token is supplied, the token is ignored and the request is processed as if the default user that is defined in the UDDI node settings made the request.

Typically, UDDI registries are configured to require authorization for publish API requests.

Data type Boolean
Default True (selected)

Authorization for custody transfer:

Specifies whether authorization that uses the authInfo element is required for custody transfer API requests. This setting is relevant only if the V3SOAP_CustodyTransfer_User_Role role is set to Everyone and WebSphere Application Server administrative security is on.

If WebSphere Application Server administrative security is off, this setting is ignored. If WebSphere Application Server administrative security is on, but the V3SOAP_CustodyTransfer_User_Role role is not set to Everyone, this setting is ignored.

If this option is selected, an authorization token is required to complete the request. If this option is cleared, an authorization token is not required. If this option is cleared and an authorization token is supplied, the token is ignored and the request is processed as if the default user that is defined in the UDDI node settings made the request.

Typically, UDDI registries are configured to require authorization for custody transfer API requests.

Data type Boolean
Default True (selected)

UDDI user policy settings:

You can view or edit the user policy settings for a UDDI registry node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **User policies**.

Allow transfer of ownership:

Specifies whether data ownership can be transferred between owners in the UDDI node. When this option is selected, data ownership can be transferred.

Data type	Boolean
Default	True (selected)

UDDI data custody policy settings:

You can view or edit the data custody settings for a UDDI registry node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Data custody policies**.

Transfer token expiration period:

Specifies length of time, in minutes, allowed before a transfer token is no longer valid.

If you set this value too high, you might expose the UDDI registry to a risk of misuse.

Data type	Integer
Default	1440
Range	1 to 2147483647 (for all intents and purposes, unlimited)

UDDI value set policy settings:

You can view or edit the value set policy settings for a UDDI registry node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Value set policies**.

Enable checked value sets:

Specifies whether checked value sets are supported. If you clear this option, publish requests of value set tModel entities that contain a checked keyedReference are rejected.

Data type	Boolean
Default	True (selected)

UDDI node miscellaneous settings:

You can view and edit settings for a UDDI node.

To view this administrative console page, click **UDDI** > **UDDI Nodes** > ***UDDI_node_id*** > **Miscellaneous policies**.

Node generates discoveryURLs:

Specifies whether a UDDI node can establish a policy on whether it generates discoveryURL URLs.

Data type	Boolean
Default	False (cleared)

Node supports HTTP Get Service:

Specifies whether the UDDI node supports an HTTP GET service for access to the XML representations of UDDI data structures.

Data type	Boolean
Default	True (selected)

URL prefix for V3 GET servlet:

Specifies the prefix for the URL to the Version 3 GET servlet that retrieves the XML representation of a published entity. This property applies to UDDI Version 3 API requests only.

The format of the prefix is http://hostname:port/uddiv3soap/, where uddiv3soap is the context root of the UDDI Version 3 SOAP servlet.

When a businessEntity entity is published, if **Node generates discovery URLs** is selected, the discoveryURL value is generated based on this prefix value. Otherwise, the discoveryURL value is empty.

The UDDI Version 3 specification recommends that you do not enable generation of discoveryURL URLs because they can affect the use of digital signatures. If you do enable generation of discoveryURL URLs, do not change the URL prefix later. Otherwise discoveryURL URLs that were generated using the earlier URL prefix no longer work.

Data type	URL
Default	http://localhost:9080/uddiv3soap/

UDDI registry administrative (JMX) interface

You can use the UDDI registry administrative interface to inspect and manage the runtime configuration of a UDDI application. You can manage the information about a UDDI node and its activation state, update properties and policies, set publish tier limits, register UDDI publishers, and control value set support.

You can read and invoke the operations of the UDDI registry administrative interface by using standard Java Management Extensions (JMX) interfaces. For more information about using JMX administrative programs, see the related links.

Each WebSphere Application Server UDDI registry application registers an MBean with an MBean identifier of UddiNode. Client applications can use this MBean to inspect and manage the runtime configuration of a UDDI application. Management activities include managing the activation state of a UDDI node, managing information about a UDDI node, updating properties and policies, setting publish tier limits, registering UDDI publishers, and controlling value set support.

You can read and invoke the UddiNode attributes and operations by using standard JMX interfaces. A UddiNodeProxy.java client utility class provides a ready-made application to connect to a UddiNode MBean and perform all the available operations. Example classes are also provided to drive UddiNodeProxy and demonstrate how to use the various UDDI management data types.

When WebSphere Application Server security is enabled, you can invoke the operations of the UddiNode MBean only if you are a user in an administrative role. Operations that make updates require the administrator or operator role. Administrator, operator, configurator and monitor roles can perform get operations.

Management of UDDI node states and attributes

You can use the UDDI registry administrative interface to manage the information about a UDDI node and its activation state.

UDDI nodes can be in one of several states, depending on the way the UDDI application is installed. The UddiNode MBean provides four read-only attributes:

- nodeID
- nodeState
- nodeDescription
- nodeApplicationName

The following MBean operations change the state of the UDDI node:

- activateNode
- deactivateNode
- initNode

nodeID

The node ID is the unique identifier for a UDDI node. If the UDDI application is installed as a default configuration, the node ID is generated automatically. If the UDDI application is installed manually, the administrator sets the node ID. The node ID must be a valid UDDI key.

```
String nodeID = uddiNode.getNode();

System.out.println("node ID: " + nodeId);
```

nodeState

The nodeState attribute can have one of the values in the following table:

Table 333. nodeState attribute values. The table lists different nodeState values along with the English text associated with each one.

nodeState value	English text associated with state
node.state.uninitialized	Not initialized
node.state.initialized	Initialized
node.state.initPending	Initialization pending
node.state.initInProgress	Initialization in progress
node.state.initMigrationPending	Migration pending
node.state.initMigration	Migration in progress
node.state.initValueSetCreationPending	Value set creation pending
node.state.initValueSetCreation	Value set creation in progress
node.state.activated	Activated
node.state.deactivated	Deactivated
node.state.unknown	Unknown

After a UDDI application is installed as a default configuration,, the UDDI node is in activated state, that is, ready to receive and process UDDI API requests. The node ID, root key generator, and some other properties are generated and you cannot change them.

After a UDDI application is installed manually, for example, because you want to specify the UDDI node ID and root key generator values, when the UDDI application starts, the UDDI node is in initPending state. In this state, you can update all writable values until you invoke the initNode

operation. The initNode operation loads base tModel entities and value set data, and writes all the configuration data to the database for the UDDI node. During initialization, the UDDI node is in initInProgress state. When initialization completes, the state changes momentarily to initialized and settles at activated. At this point, the state can be switched only between the activated and deactivated states, using the deactivateNode and activateNode MBean operations.

Each node state value is a message key that can be looked up in the messages.properties resource bundle. You can retrieve the attribute value by using the getNodeState method of the UddiNodeProxy class.

1. Invoke the getNodeState method:

```
String nodeStateKey = uddiNode.getNodeState();
```

2. Look up translated text from the resource bundle and produce it as output:

```
String messages = "com.ibm.uddi.v3.management.messages";

ResourceBundle bundle = ResourceBundle.getBundle(messages,Locale.ENGLISH);

String nodeStateText =  bundle.getString(nodeStateKey);

System.out.println("node state: " + nodeStateText);
```

nodeDescription

You can get the administrator-assigned description for the UDDI node by using the getNodeDescription method of the UddiNodeProxy class.

- Invoke the getNodeDescription method and produce output:

```
String nodeDescription = uddiNode.getNodeDescription();
System.out.println("node description: " + nodeDescription);
```

nodeApplicationName

You can use the nodeApplicationName attribute to discover where the UDDI application that corresponds to the UDDI node is installed. The value is a concatenation of the cell, node, and server names, separated by colons. To retrieve the application location, use the getApplicationId method of the UddiNodeProxy class.

- Invoke the getApplicationId method and produce output:

```
String nodeApplicationId = uddiNode.getApplicationId();

System.out.println("node application location: " + nodeApplicationId);
```

activateNode

Changes the state of the UDDI node to activated, if the UDDI node is currently deactivated.

- Invoke the activateNode operation:

```
uddiNode.activateNode();
```

deactivateNode

Changes the state of the UDDI node to deactivated, if the UDDI node is currently activated.

- Invoke the deactivateNode operation:

```
uddiNode.deactivateNode();
```

initNode

Initializes the UDDI node. When initialization completes, the UDDI node is in the activated state.

- Invoke the initNode operation:

```
 uddiNode.initNode();
```

Management of UDDI node configuration properties

You can use the UDDI registry administrative interface to manage the UDDI node runtime behavior by setting the configuration properties.

UDDI node runtime behavior is affected by the setting of several configuration properties. The UddiNode MBean provides the following operations to inspect and update the configuration properties:

- getProperties

- getProperty
- updateProperty
- updateProperties

In the samples for WebSphere Application Server, the ManagePropertiesSample class in the UDDI registry samples demonstrates these operations.

getProperties

Returns a collection of all configuration properties as ConfigurationProperty objects.

1. Invoke the getProperties operation:

```
List properties = uddiNode.getProperties();
```

2. Cast each collection member to the ConfigurationProperty object:

```
 if (properties != null) {
    for (Iterator iter = properties.iterator(); iter.hasNext();) {
       ConfigurationProperty property = (ConfigurationProperty) iter.next();
       System.out.println(property);
    }
 }
```

When you have the ConfigurationProperty objects, you can inspect attributes such as the ID, value, and type. You can determine whether the property is read-only or required for initialization, and get the name and description message keys. For example, if you invoke the toString method, results similar to the following example are returned:

```
ConfigurationProperty
id: operatorNodeIDValue
nameKey: property.name.operatorNodeIDValue
descriptionKey: property.desc.operatorNodeIDValue
type: java.lang.String
value: uddi:capnscarlet:capnscarlet:server1:default
unitsKey:
readOnly: true
required: true
usingMessageKeys: false
validValues: none
```

You can use the nameKey and descriptionKey values to look up the translated name and description for a given locale, using the messages.properties resource in the sample package.

getProperty

Returns the ConfigurationProperty object with the specified ID. Available property IDs are specified in PropertyConstants with descriptions of the purpose of the corresponding properties.

1. Invoke the getProperty operation:

```
ConfigurationProperty property =
uddiNode.getProperty(PropertyConstants.DATABASE_MAX_RESULT_COUNT);
```

2. To retrieve the value of the property, you can use the getValue method, which returns an Object, but in this case, the property is an integer type, so it is easier to retrieve the value by using the convenience method getIntegerValue:

```
int maxResults = property.getIntegerValue();
```

updateProperty

Updates the value of the ConfigurationProperty object with the specified ID. Available property IDs are specified in PropertyConstants with descriptions of the purpose of the corresponding properties. Although you can invoke the setter methods in a ConfigurationProperty object, the only value that is updated in the UDDI node is the value. To update a property, typically, use the following steps:

1. Create a ConfigurationProperty object and set its ID:

```
ConfigurationProperty defaultLanguage = new ConfigurationProperty();
defaultLanguage.setId(PropertyConstants.DEFAULT_LANGUAGE);
```

2. Set the value:

```
defaultLanguage.setStringValue("ja");
```

3. Invoke the updateProperty operation:

```
uddiNode.updateProperty(defaultLanguage);
```

updateProperties

Updates several ConfigurationProperty objects in a single request. Set up the ConfigurationProperty objects in the same way as for the updateProperty operation.

1. Add the updated properties to a list:

```
List updatedProperties = new ArrayList();

updatedProperties.add(updatedProperty1);
updatedProperties.add(updatedProperty2);
```

2. Invoke the updateProperties operation:

```
uddiNode.updateProperties(updatedProperties);
```

Management of UDDI node policies

You can use the UDDI registry administrative interface to manage policies that affect the UDDI API.

The UddiNode MBean provides the following operations to manage the policies that affect the behavior of the UDDI API:

- getPolicyGroups
- getPolicyGroup
- getPolicy
- updatePolicy
- updatePolicies

In the samples for WebSphere Application Server, the ManagePoliciesSample class in the UDDI registry samples demonstrates these operations.

getPolicyGroups

Returns a collection of all the policy groups as PolicyGroup objects.

1. Invoke the getPolicyGroups operation:

```
List policyGroups = uddiNode.getPolicyGroups();
```

2. Cast each collection member to PolicyGroup:

```
if (policyGroups != null) {
for (Iterator iter = policyGroups.iterator(); iter.hasNext();) {
   PolicyGroup policyGroup = (PolicyGroup) iter.next();
   System.out.println(policyGroup);
   }
}
```

Each policy group has an ID, name, and description key, which you can look up in the messages.properties resource in the sample package. Although the PolicyGroup class does have a getPolicies method, PolicyGroup objects that are returned by the getPolicyGroups operation do not contain any Policy objects. Because of this behavior, clients can determine the known policy groups, and their IDs, without retrieving the entire set of policies in one request. To retrieve the policies in a policy group, use the getPolicyGroup operation.

getPolicyGroup

Returns the PolicyGroup object with the supplied ID.

1. Convert the policy group ID to a string:

```
String groupId = Integer.toString(PolicyConstants.REG_APIS_GROUP);
```

2. Invoke the getPolicyGroup operation:

```
PolicyGroup policyGroup = uddiNode.getPolicyGroup(groupId);
```

getPolicy

Returns the Policy object for the specified ID. As with a configuration property, a Policy object has an ID, name and description keys, type, value, and indicators that specify whether the policy is read-only or required for node initialization.

1. Convert the policy ID to a string:

```
String policyId = Integer.toString(
   PolicyConstants.REG_AUTHORIZATION_FOR_INQUIRY_API);
```

2. Invoke the getPolicy operation:

```
Policy policy = uddiNode.getPolicy(policyId);
```

updatePolicy

Updates the value of the Policy object with the specified ID. Available policy IDs are specified in PolicyConstants with descriptions of the purpose of the corresponding policies. Although you can invoke the setter methods in a Policy object, the only value that is updated in the UDDI node is the value. To update a policy, typically, use the following steps:

1. Create a Policy object and set its ID:

```
Policy updatedPolicy = new Policy();
String policyId = Integer.toString(PolicyConstants.REG_SUPPORTS_UUID_KEYS);
updatedPolicy.setId(policyId);
```

2. Set the value:

```
updatedPolicy.setBooleanValue(true);
```

3. Invoke the updatePolicy operation:

```
uddiNode.updatePolicy(updatedPolicy);
```

updatePolicies

Updates several Policy objects in a single request. Set up the Policy objects in the same way as for the updatePolicy operation.

1. Add updated policies to a list:

```
List updatedPolicies = new ArrayList();

updatedPolicies.add(updatedPolicy1);
updatedPolicies.add(updatedPolicy2);
```

2. Invoke the updatePolicies operation:

```
uddiNode.updatePolicies(updatedPolicies);
```

Management of UDDI node tiers

You can use the UDDI registry administrative interface to set publish tier limits, which control the number of each type of UDDI entity that a publisher can save in the UDDI registry.

A tier has an ID, an administrator-defined name and description, and a set of limits, one for each type of entity. The UddiNode MBean provides the following operations to manage tiers:

- createTier
- getTierDetail
- getTierInfos
- getLimitInfos
- setDefaultTier
- updateTier
- deleteTier
- getUserCount

In the samples for WebSphere Application Server, the ManageTiersSample class in the UDDI registry samples demonstrates these operations.

createTier

Creates a new tier, with specified publish limits for each UDDI entity.

1. Set the tier name and description in a TierInfo object.

```
String tierName = "Tier 100";
String tierDescription = "A tier with all limits set to 100.";

TierInfo tierInfo = new TierInfo(null, tierName, tierDescription);
```

2. Define Limit objects for each UDDI entity:

```
List limits = new ArrayList();

Limit businessLimit = new Limit();
businessLimit.setIntegerValue(100);

businessLimit.setId(LimitConstants.BUSINESS_LIMIT);

Limit serviceLimit = new Limit();
serviceLimit.setIntegerValue(100);
serviceLimit.setId(LimitConstants.SERVICE_LIMIT);

Limit bindingLimit = new Limit();
bindingLimit.setIntegerValue(100);
bindingLimit.setId(LimitConstants.BINDING_LIMIT);

Limit tModelLimit = new Limit();
tModelLimit.setIntegerValue(100);
tModelLimit.setId(LimitConstants.TMODEL_LIMIT);

Limit assertionLimit = new Limit();
assertionLimit.setIntegerValue(100);

assertionLimit.setId(LimitConstants.ASSERTION_LIMIT);
limits.add(businessLimit);
limits.add(serviceLimit);
limits.add(bindingLimit);
limits.add(tModelLimit);
limits.add(assertionLimit);
```

3. Create the Tier object:

```
Tier tier = new Tier(tierInfo, limits);
```

4. Invoke the createTier operation and retrieve the created tier:

```
Tier createdTier = uddiNode.createTier(tier);
```

5. Inspect the generated tier ID of the created tier:

```
tierId = createdTier.getId();
System.out.println("created tier has ID: " + tierId);
```

getTierDetail

Returns the Tier object for the given tier ID. The Tier class has getter methods for the tier ID, tier name and description, as set by the administrator, and the collection of Limit objects, which specify how many of each UDDI entity type UDDI publishers that are allocated to the tier can publish. The isDefault method indicates whether the tier is the default tier, that is, the tier that is allocated to UDDI publishers when auto registration is enabled.

- Invoke the getTierDetail operation:

```
Tier tier = uddiNode.getTierDetail("2");
```

updateTier

Updates the tier contents with the supplied Tier object.

1. Update an existing Tier object, which might be newly instantiated, or returned by the getTierDetail or createTier operations. The following example retains the tier name, description, and all the limit values except the limit that is updated:

```
modifiedTier.setName(tier.getName());
modifiedTier.setDescription(tier.getDescription());

Limit tModelLimit = new Limit();
tModelLimit.setId(LimitConstants.TMODEL_LIMIT);
tModelLimit.setIntegerValue(50);

List updatedLimits = new ArrayList();
updatedLimits.add(tModelLimit);

modifiedTier.setLimits(updatedLimits);
```

2. Invoke the updateTier operation:

```
uddiNode.updateTier(modifiedTier);
```

getTierInfos

Returns a collection of lightweight tier descriptor objects (TierInfo) that contain the tier ID, tier name, tier description, and whether the tier is the default tier.

1. Invoke the getTierInfos operation:

```
List tierInfos = uddiNode.getTierInfos();
```

2. Output the content of each TierInfo object:

```
if (tierInfos != null) {

   for (Iterator iter = tierInfos.iterator(); iter.hasNext();) {
      TierInfo tierInfo = (TierInfo) iter.next();
      System.out.println(tierInfo);
   }
}
```

setDefaultTier

Specifies that the tier with the given tier ID is the default tier. The default tier is the tier that is allocated to UDDI publishers when auto registration is enabled. Typically, you set this to a tier with low publish limits to stop casual users from publishing too many entities.

- Invoke the setDefaultTier operation:

```
uddiNode.setDefaultTier("4");
```

deleteTier

Removes the tier with the given tier ID. Tiers can be removed only if they have no UDDI publishers assigned to them, and the tier is not the default tier.

1. Invoke the deleteTier operation:

```
uddiNode.deleteTier("4");
```

getUserCount

Returns the number of UDDI publishers that are assigned to the tier that is specified by the tier ID.

- Invoke the getUserCount operation:

```
Integer userCount = uddiNode.getUserCount("4");
System.out.println("users in tier 4: " + userCount.intValue());
```

getLimitInfos

Returns a collection of Limit objects that represent the limit values for each type of UDDI entity. Limits are used in Tier objects.

getLimitInfos

Returns collection of Limit objects representing the limit values for each type of UDDI entity. Limits are used in Tier objects.

1. Invoke the getLimitInfos operation:

```
List limits = uddiNode.getLimitInfos();
```

2. Output the ID and limit value for each Limit object:

```
for (Iterator iter = limits.iterator(); iter.hasNext();) {
   Limit limit = (Limit) iter.next();

   System.out.println("limit ID: " + limit.getId() + ", limit value: "
                      + limit.getIntegerValue());
}
```

Management of UDDI publishers

You can use the UDDI registry administrative interface to register UDDI publishers.

Managing UDDI publishers

The UddiNode MBean provides the following operations to manage UDDI publishers:

- createUddiUser
- createUddiUsers
- updateUddiUser
- deleteUddiUser

- getUddiUser
- getUserInfos
- getEntitlementInfos
- assignTier
- getUserTier
-

In the samples for WebSphere Application Server, the ManagePublishersSample class in the UDDI registry samples demonstrates these operations. An example is provided for each, making use of the UddiNodeProxy client class.

createUddiUser

Registers a single UDDI publisher in a specified tier with specified entitlements. The UddiUser class represents the UDDI publisher, and is constructed using a user ID, a TierInfo object that specifies the tier ID to allocate the UDDI publisher to, and a collection of Entitlement objects that specify what the UDDI publisher is permitted to do.

Tip: To allocate the UDDI publisher default entitlements, set the entitlements parameter to null.

1. Create the UddiUser object:

```
UddiUser user = new UddiUser("user1", new TierInfo("3"), null);
```

2. Invoke the createUddiUser operation:

```
uddiNode.createUddiUser(user);
```

createUddiUsers

Registers multiple UDDI publishers. The following example shows how to register seven UDDI publishers with default entitlements in one call.

1. Create TierInfo objects for the tiers that the publishers are allocated to:

```
TierInfo tier1 = new TierInfo("1");
TierInfo tier4 = new TierInfo("4");
```

2. Create UddiUser objects for each UDDI publisher, specifying the tier for each publisher:

```
UddiUser publisher1 = new UddiUser("Publisher1", tier4, null);
UddiUser publisher2 = new UddiUser("Publisher2", tier4, null);
UddiUser publisher3 = new UddiUser("Publisher3", tier4, null);
UddiUser publisher4 = new UddiUser("Publisher4", tier1, null);
UddiUser publisher5 = new UddiUser("Publisher5", tier1, null);
UddiUser cts1 = new UddiUser("cts1", tier4, null);
UddiUser cts2 = new UddiUser("cts2", tier4, null);
```

3. Add the UddiUser objects to a list:

```
List uddiUsers = new ArrayList();

uddiUsers.add(publisher1);
uddiUsers.add(publisher2);
uddiUsers.add(publisher3);
uddiUsers.add(publisher4);
uddiUsers.add(publisher5);
uddiUsers.add(cts1);
uddiUsers.add(cts2);
```

4. Invoke the createUddiUsers operation:

```
uddiNode.createUddiUsers(uddiUsers);
```

updateUddiUser

Updates a UDDI publisher with the details in the supplied UddiUser object. Typically, you use this operation to change the tier of one UDDI publisher or to update the entitlements of a UDDI publisher. Supply only the entitlements that you want to update; other available entitlements retain their existing values.

1. Create Entitlement objects with the appropriate permission. The entitlement IDs are found in EntitlementConstants.

```
Entitlement publishUuiDKeyGenerator =
   new Entitlement(PUBLISH_UUID_KEY_GENERATOR, true);
Entitlement publishWithUuidKey =
   new Entitlement(PUBLISH_WITH_UUID_KEY, true);
```

2. Add the Entitlement objects to a list:

```
List entitlements = new ArrayList();
entitlements.add(publishUuiDKeyGenerator);
entitlements.add(publishWithUuidKey);
```

3. Update a UddiUser object with the updated entitlements:

```
user.setEntitlements(entitlements);
```

4. Invoke the updateUddiUser operation:

```
uddiNode.updateUddiUser(user);
```

getUddiUser

Retrieves details about a UDDI publisher in the form of a UddiUser object. This specifies the UDDI publisher ID, information about the tier the UDDI publisher is assigned to and the entitlements of the UDDI publisher.

1. Invoke the getUddiUser operation:

```
UddiUser user1 = uddiNode.getUddiUser("user1");
```

2. Output the contents of the UddiUser object:

```
System.out.println("retrieved user: " + user1);
```

getUserInfos

Returns a collection of UserInfo objects. Each UserInfo object represents a UDDI publisher that is known to the UDDI node, and the tier that the UDDI publisher is allocated to. To get more details about a specific UDDI publisher, including the tier ID and entitlements, use the getUddiUser operation.

1. Invoke the getUserInfos operation:

```
List registeredUsers = uddiNode.getUserInfos();
```

2. Output the UserInfo objects:

```
System.out.println("retrieved registered users: ");
System.out.println(registeredUsers);
```

getEntitlementInfos

Returns a collection of Entitlement objects. Each entitlement is a property that controls whether a UDDI publisher has permission to undertake a specified action.

1. Invoke the getEntitlementInfos operation:

```
List entitlementInfos = uddiNode.getEntitlementInfos();
```

2. Specify where to find message resources:

```
String messages = "com.ibm.uddi.v3.management.messages";
ResourceBundle bundle = ResourceBundle.getBundle(messages, Locale.ENGLISH);
```

3. Iterate through the Entitlement objects, displaying the ID, name, and description:

```
for (Iterator iter = entitlementInfos.iterator(); iter.hasNext();) {
   Entitlement entitlement = (Entitlement) iter.next();

   StringBuffer entitlementOutput = new StringBuffer();

   String entitlementId = entitlement.getId();
   String entitlementName = bundle.getString(entitlement.getNameKey());
   String entitlementDescription =
                    bundle.getString(entitlement.getDescriptionKey());

   entitlementOutput.append("Entitlement id: ");
   entitlementOutput.append(entitlementId);
   entitlementOutput.append("\n  name: ");
   entitlementOutput.append(entitlementName);
   entitlementOutput.append("\n  description: ");
   entitlementOutput.append(entitlementDescription);

   System.out.println(entitlementOutput.toString());
}
```

deleteUddiUser

Removes the UDDI publisher with the specified user ID from the UDDI registry.

- Invoke the deleteUddiUser operation:

```
uddiNode.deleteUddiUser("user1");
```

assignTier

Assigns the UDDI publishers with the supplied IDs to the specified tier. This operationis useful when you want to restrict several UDDI publishers, for example by assigning them to a tier that does not allow publishing of any entities.

1. Create a list of publisher IDs:

```
List uddiUserIds = new ArrayList();

uddiUserIds.add("Publisher1");
uddiUserIds.add("Publisher2");
uddiUserIds.add("Publisher3");
uddiUserIds.add("Publisher4");
uddiUserIds.add("Publisher5");
uddiUserIds.add("cts1");
uddiUserIds.add("cts2");
```

2. Invoke the assignTier operation:

```
uddiNode.assignTier(uddiUserIds, "0");
```

getUserTier

Returns information about the tier that a UDDI publisher is assigned to. The returned TierInfo has getter methods for retrieving the tier ID, tier name, tier description, and whether the tier is the default tier.

1. Invoke the getUserTier operation:

```
TierInfo tierInfo = getUserTier("Publisher3");
```

2. Output the contents of the TierInfo object:

```
System.out.println(tierInfo);
```

Management of UDDI node value sets

You can use the UDDI registry administrative interface to inspect and manage the runtime configuration of a UDDI application. You can manage the information about a UDDI node and its activation state, update properties and policies, set publish tier limits, register UDDI publishers, and control value set support.

Value sets are represented in a UDDI registry as value set tModel entities, with a keyedReference UDDI type with the value categorization. Such value sets are backed with a set of valid values. For user-defined value sets, this data is loaded into the UDDI registry using UddiNode MBean operations, although it is more convenient to do this using the User defined value set tool.

Each value set can be controlled by policy as being supported or not supported. When a value set is supported by policy, it can be referenced in UDDI publish requests. The UddiNode MBean provides the following operations to manage value sets and their data:

- getValueSets
- getValueSetDetail
- getValueSetProperty
- updateValueSet
- updateValueSets
- loadValueSet
- changeValueSetTModelKey
- unloadValueSet
- isExistingValueSet

In the samples for WebSphere Application Server, the ManageValueSetsSample class in the UDDI registry samples demonstrates these operations.

getValueSets

Returns a collection of ValueSetStatus objects.

1. Invoke the getValueSets operation:

```
List valueSets = uddiNode.getValueSets();
```

2. Cast each element to ValueSetStatus and output contents:

```
for (Iterator iter = valueSets.iterator(); iter.hasNext();) {

   ValueSetStatus valueSetStatus = (ValueSetStatus) iter.next();
   System.out.println(valueSetStatus);
}
```

getValueSetDetail

Returns a ValueSetStatus object for the given value set tModel key.

1. Invoke the getValueSetDetail operation:

```
uddiNode.getValueSetDetail("uddi:uddi.org:ubr:categorization:naics:2002");
```

2. Retrieve and display the details:

```
String name = valueSetStatus.getName();
String displayName = valueSetStatus.getDisplayName();
boolean supported = valueSetStatus.isSupported();

System.out.println("name: " + name);
System.out.println("display name: " + displayName);
System.out.println("supported: " + supported);
```

3. Display the value set properties:

```
List properties = valueSetStatus.getProperties();

for (Iterator iter = properties.iterator(); iter.hasNext();) {

   ValueSetProperty property = (ValueSetProperty) iter.next();
   System.out.println(property);
}
```

getValueSetProperty

Returns a property of a value set as a ValueSetProperty object. This operation is mainly for the administrative console to render properties of a value set as a row in a table. For example, one such property is the keyedReference property, which indicates whether the value set is checked.

1. Invoke the getValueSetProperty operation:

```
uddiNode.getValueSetProperty("uddi:uddi.org:ubr:categorization:naics:2002",
   ValueSetPropertyConstants.VS_CHECKED);
```

2. Read and display the boolean value of the property:

```
boolean checked = valueSetProperty.getBooleanValue();

System.out.println("checked: " + checked);
```

updateValueSet

Updates the value set status. Only the supported attribute can be updated. All other setter methods are used by the UDDI application.

1. Create a ValueSetStatus object specifying the tModel key and the updated supported value:

```
ValueSetStatus updatedStatus = new ValueSetStatus();
updatedStatus.setTModelKey("uddi:uddi.org:ubr:categorization:naics:2002");
updatedStatus.setSupported(true);
```

2. Invoke the updateValueSet operation:

```
uddiNode.updateValueSet(updatedStatus);
```

updateValueSets

Updates the value set status for multiple value sets. Similarly to the updateValueSet operation, only the supported attribute is updated.

1. Populate the list with updated ValueSetStatus objects:

```
List valueSets = new ArrayList();

ValueSetStatus valueSetStatus = new ValueSetStatus();
valueSetStatus.setTModelKey("uddi:uddi.org:ubr:categorization:naics:2002");
```

```
valueSetStatus.setSupported(false);
valueSets.add(valueSetStatus);

valueSetStatus = new ValueSetStatus();
valueSetStatus.setTModelKey("uddi:uddi.org:ubr:categorizationgroup:wgs84");
valueSetStatus.setSupported(false);
valueSets.add(valueSetStatus);

valueSetStatus = new ValueSetStatus();
valueSetStatus.setTModelKey("uddi:uddi.org:ubr:identifier:iso6523:icd");
valueSetStatus.setSupported(false);
valueSets.add(valueSetStatus);
```

2. Invoke the updateValueSets operation:

```
uddiNode.updateValueSets(valueSets);
```

loadValueSet

Loads values for a value set from a UDDI registry Version 3 or Version 2 taxonomy data file on the local file system.

Note: There is also a loadValueSet operation that takes a ValueSetData object, but this is only for use by the user-defined value set tool.

1. Invoke the loadValueSet operation:

```
uddiNode.loadValueSet("/valuesets/myvalueset.txt",
   "uddi:cell:node:server:myValueSet");
```

changeValueSetTModelKey

Allocate any value set values that are allocated to one value set tModel to a new value set tModel.

- Invoke the changeValueSetTModelKey operation, specifying old and new tModel keys:

```
uddiNode.changeValueSetTModelKey(
"uddi:cell:node:server:myValueSet",
"uddi:cell:node:server:myNewValueSet");
```

unloadValueSet

Unloads values for a value set with the given tModel key.

- Invoke the unloadValueSet operation:

```
uddiNode.unloadValueSet("uddi:myValueSet");
```

isExistingValueSet

Determines whether value set data exists for the given tModel key.

1. Invoke the isExistingValueSet operation and display the result:

```
boolean exists = uddiNode.isExistingValueSet(
    "uddi:uddi.org:ubr:categorization:naics:2002");
System.out.println("NAICS 2002 is a value set: " + exists);
```

User-defined value set support in the UDDI registry

You can define multiple value sets and add custom value sets to the UDDI Version 3 registry. In UDDI Version 2, this feature was called custom taxonomy support.

The UDDI Version 3 registry provides the structure and modeling tools to find information in a registry effectively. Also, the verification of data in a UDDI registry is crucial to its mission of description, discovery and integration.

You can define multiple value sets to use UDDI. Therefore, multiple classification schemes can be overlaid on a single UDDI entity. Organizations can use this capability to extend the set of such systems that UDDI registries support. You are not restricted to a single system, but can employ several different classification systems simultaneously.

Default value sets are shipped with the product. The UDDI Version 3 registry provides tools to add custom, or user-defined, value sets. You can use such value sets to categorize UDDI entities more specifically when they are published, which enhances the capability of client to find specific data.

User-defined value sets can be either checked or unchecked. This is indicated by a keyedReference element in the categoryBag element of the tModel entity that represents a value set (a categorization tModel). These keyedReference elements have the tModel key for uddi-org:types and are added to the categoryBag to further describe the behavior of the categorization tModel, as follows:

checked
: Marking a tModel entity with this classification asserts that the entity represents a categorization, identifier, or namespace tModel entity that has a validation service to check that category values are present in a specified value set.

unchecked
: Marking a tModel entity with this classification asserts that the entity represents a categorization, identifier, or namespace tModel entity that does not have a validation service.

The following procedure describes how to add user-defined value sets, and display their allowed values in the UDDI user console value set tree display. Rational Application Developer has a Web Services Explorer user interface that also allows addition and display of custom checked value sets. The publisher of a value set categorization tModel entity might specify a display name for use in UDDI user console implementations.

Adding a user-defined value set

To add a user-defined value set to the UDDI registry, use the following procedure:

1. Publish a categorization tModel entity.
2. Load the user-defined value set data.
3. Enable support for the value set by using the administrative console.

To enable support for the value set, you must be a user in an administrative role. This means that you cannot add user-defined value sets to the UDDI registry without administrator permission.

The checked value set is referenced only when all steps of the procedure are complete. Value set data must be provided for validating checked value sets.

User consoles might use value set data for unchecked value sets, but it is not a requirement, and is usually used only for presentation of deprecated value sets, such as unspc-org:unspc, and for compatibility with earlier versions.

If the value set is checked, any publish requests that have a categoryBag element that contains keyedReference elements with the new categorization tModel are validated. If there is value set data corresponding to the categorization tModel entity in the registry database, only valid values are accepted. If there is no value set data in the database, all values are rejected and the publish request fails. If the categorization tModel entity is unchecked, all values are allowed, regardless of whether a corresponding value set is present in the UDDI registry database. The value set tModel entity is not available for use until the administrator enables support for it by using the administrative console or the JMX interface.

The following procedure provides a suggested approach to add a user-defined value set to the UDDI registry:

1. Publish the categorization tModel entity with the following values:

keyedReference element	**values**
uddi-org:categorization:types	keyValue = categorization
uddi-org:categorization:types	keyName = Checked value set and keyValue = checked or keyName = Unchecked value set and keyValue = unchecked

keyedReference element	values
uddi-org:categorization:general_keywords	supply the value set display name

2. Load the user-defined value set data into the UDDI registry database using the UDDIUserDefinedValueSet utility.
3. Use the administrative console to set the status of the value set to supported in the Value set settings. Alternatively, you can do this directly by using the JMX interface.

The SOAP and Enterprise JavaBeans (EJB) interfaces can use categorization tModel entities as soon as they are published. However, for the UDDI registry user console, the UDDI application must be restarted, because the console gathers the list of categorizations to use in the value set tree display when the application starts.

Publish a checked categorization tModel entity

You publish a checked categorization tModel entity as the first step in the procedure to add a user-defined value set to the UDDI registry.

This topic describes how to publish a checked categorization tModel entity with the keyName “Checked value set” for a user-defined value set to use.

Publish a tModel entity to the UDDI registry with a categoryBag element that contains the keyedReference elements shown in the following table.

Table 334. keyedReference elements for a checked categorization tModel entity. The table lists different tModelKey elements along with their KeyName, KeyValue and additional user notes.

tModelKey	KeyName	KeyValue	Notes
uddi:uddi-org:categorization:types	categorization	categorization	To choose this tModelKey in the UDDI registry user interface, select the category type **UDDI Types**. This element indicates that this tModel entity is a categorization tModel entity (required).
uddi:uddi-org:categorization:types	Checked value set	checked	To choose this tModelKey in the UDDI registry user interface, select the category type **UDDI Types**. This element indicates that use of the tModel entity is checked against a list of valid data (required). If this keyedReference element is omitted, or a value of unchecked is specified, this indicates that this categorization is unchecked.
uddi:uddi-org:categorization:general_keywords	urn:x-ibm:uddi:customTaxonomy:displayName	*user-defined Value Set displayName*	To choose this tModelKey in the UDDI registry user interface, select the category type **categorization: general_keywords**. This element indicates special use of the general keywords value set, with a proprietary uniform resource name (URN) as the keyName value, defines a name for the user-defined value set that is intended for use in user console implementations where the full tModel name might be too long. The value can be 1-255 characters (inclusive) long.

You can use the displayName value to give a value set a label that is meaningful to users, that is greater than 8 characters, and that is different to the published tModelName, which might be as long as 255 characters. The UDDI user console can display this label in a value set tree or in a drop-down list of available value sets. The following diagram shows an example:

Figure 50. Use of the display name in the UDDI user console

It is advisable to specify a unique name for urn:x-ibm:customTaxonomy:displayName to avoid confusion when it is displayed in user interfaces.

To publish a new categorization tModel that uses SOAP, use the following message:

```
<save_tModel generic="3.0" xmlns="urn:uddi-org:api_v3">
  <authInfo></authInfo>>
  <tModel tModelKey="">
    <name>Natural Foods tModel</name>
    <categoryBag>
      <keyedReference tModelKey="uddi:uddi.org:categorization:types"
        keyName="categorization" keyValue="categorization"/>
      <keyedReference tModelKey="uddi:uddi.org:categorization:types"
        keyName="Checked value set" keyValue="checked"/>
      <keyedReference tModelKey="uddi:uddi.org:categorization:general_keywords"
        keyName="urn:x-ibm:uddi:customTaxonomy:displayName"
        keyValue="Natural Foods"/>
    </categoryBag>
  </tModel>
</save_tModel>
```

Tip: To specify an unchecked categorization, change the keyName value from “Checked value set” to “Unchecked value set” and change the keyValue value from checked to unchecked. Alternatively, omit that keyedReference element completely.

Load user-defined value set data

You can use the UDDIUserDefinedValueSet utility to load value set data into the UDDI registry, assign existing value set data to another tModel entity, and unload existing value set data. You load value set data as a step in the procedure to add a user-defined value set to the UDDI registry.

Format of the value set data file

Value set data is identified by a unique code value, an optional description, and a parent code that specifies its relationship with other code values. Value set data must adhere to this format.

You must save the file in UTF-8 format.

Table 335. Value set data format. The table lists the different value set column names, shows their maximum character lengths and provides a description of each one.

Column name	Maximum length	Description of use
code	765	Unique value in the value set, which is used for validation
description	765	Typically used by UDDI user consoles and optionally used in the keyedReference element as the keyName value
parentcode	765	Indicates the existing code that is the logical parent of this code, and is used in tree displays

Typically, columns are delimited in the value set data file by number (#) characters, as shown in the following example:

```
00#Food#00
10#Fruit#00
101#Apples#10
102#Oranges#10
103#Pears#10
```

```
1031#Anjou#103
1032#Conference#103
1033#Bosc#103
104#Pomegranates#10
20#Vegetables#00
201#Carrots#20
202#Potatoes#20
203#Peas#20
204#Sprouts#20
```

In the example, Food is the description of the root node and it has the child nodes Fruit and Vegetables; the parentcode values of both these child nodes are the same as the code value of Food.

The value set data in the example file can then be rendered in a tree, as shown in the following example:

```
Food
  Fruit
    Apples
    Oranges
    Pears
      Anjou
      Conference
      Bosc
    Pomegranates
  Vegetables
    Carrots
    Potatoes
    Peas
    Sprouts
```

UDDIUserDefinedValueSet utility

You can use the UDDIUserDefinedValueSet utility to load value set data into the UDDI registry, assign existing value set data to another tModel entity, and unload existing value set data. The utility also supports custom taxonomy files that are used in UDDI Version 2.

This utility uses the UDDI registry JMX interface and therefore requires a number of connection parameters.

Usage:

```
UDDIUserDefinedValueSet '{'function'}' [options]
```

Functions:

```
-load <path> <key>         Load value set data from specified file
-newKey <oldKey> <newKey>  Move value set to a new tModel
-unload <key>              Unload existing value set
```

Options:

```
-properties <path>         Specify location of configuration file
-host <host name>          Application Server host
-port <port>               SOAP Lister port number
-node <node name>          Node running a UDDI server
-server <server name>      Server with UDDI deployed
-columnDelimiter <delim>   Character delimiter to denote field end
-stringDelimiter <delim>   Character delimiter to denote strings
```

Connector security parameters

```
-userName <name>
-password <password>
-trustStore <path>
-trustStorePassword <password>
-keyStore <name>
-keyStorePassword <password>
```

Ensure that the command window that you run the UDDIUserDefinedValueSet utility from uses a suitable code page and font to display the characters that are in the value set name. If you use an incorrect code page or font, a successful load might result in unclear messages, and it might be difficult to use the unload and newKey functions.

The UDDIUserDefinedValueSet script is in the *app_server_root*/bin directory.

If you do not supply any connection parameters, a connection is sought on the local host using the default application server SOAP port number.

Command arguments are not case sensitive.

Optionally, you can use the properties parameter to specify a configuration file. This configuration file determines optional properties that you can also specify on the command line. Properties that are specified on the command line override the values in the configuration file. These properties are largely JMX connection parameters and security parameters.

Typically, you use the stringDelimiter parameter when a description value contains the same character as the column delimiter character. For example, if the columnDelimiter parameter is set to a comma (,) and there is a value set description value of “Fruits, citrus”, to include this description in the value set data file, set the stringDelimiter parameter to quotation marks (") and enclose the description in quotes, for example, `"Fruits, citrus"`. Note that you must set a backslash (\) as an escape character to show that the literal character is used.

If you attempt to load a value set to a tModel entity that has existing value set data, a warning message is displayed. To override this warning, you can use the override argument. You also require this argument when you move value set data to a new tModel entity by using the newKey function when the tModel entity is checked, and when you unload value set data for a checked tModel entity.

Table 336. Command line arguments and properties. The table lists different command line arguments and their properties and contains a comment on each one.

Command line arguments and example data	Property and example data	Comments
-columnDelimiter #	column.delimiter=#	The column delimiter that is used in value set data files.
-stringDelimiter \"	string.delimiter=\"	The field delimiter. This value must be different from the column.delimiter value.
-host ibm.com	host=ibm.com	The host name of the system that is running the application server
-port 8880	port=8880	The SOAP port number of the application server.
-node ibmNode	node=ibmNode	The name of the node that runs the server with the UDDI registry.
-server server1	server=server1	The server that runs the UDDI registry.
-userName ibmuser	security.username=ibmuser	The user name. This value is required if WebSphere Application Server security is turned on.
-password mypassword	security.password=mypassword	The password.
-trustStore /TrustStoreLocation	security.truststore=/ TrustStoreLocation	The truststore file location.
-keyStore ibmkeystore	security.keystore=ibmkeystore	The keystore name.
-trustStorepassword trustpass	security.truststore.password=trustpass	The truststore password
-keyStorePassword keypass	security.keystore.password=keypass	The keystore password.

Usage examples

Load value set data for a tModel entity on the local UDDI registry, using the percent sign as a column marker in the valuesetdata.txt file.

Move value set data from one checked tModel entity to another on a UDDI registry in a network deployment configuration.

Unload a value set from a tModel entity from a server with security enabled. Supply the connection and security parameters in the file myproperties.properties, but supply the server and password arguments on the command line. Arguments that are supplied on the command line augment or override the arguments in the properties file.

Enable support for a user-defined value set

You can enable support for the value set by using the administrative console. You do this as part of the procedure to add a user-defined value set to the UDDI registry.

To enable support for the value set, you must be a user in an administrative role. This means that you cannot add user-defined value sets to the UDDI registry without administrator permission.

Use the following steps in the administrative console:

1. Click **UDDI** > **UDDI Nodes** > ***node_name*** > **[Additional properties] Value sets**.
2. Select the value set.
3. Click **Enable Support**.

Validation and error handling for user-defined value sets

The UDDI registry user console performs validation while a save tModel entity request is built, that is, before the publish occurs.

For example, if you try to add two customTaxonomy:displayName keyedReference elements, the following message is displayed:

```
Advice: Only one 'urn:x-ibm:uddi:customTaxonomy:displayName' key name is
allowed for the 'Other' taxonomy.
```

If a keyedReference element that contains a keyName value that starts with urn:x-ibm:uddi:customTaxonomy: is followed by anything other than displayName, the following message is displayed:

```
Advice: Only key name values of 'urn:x-ibm:uddi:customTaxonomy:displayName'
are supported.
```

For requests where the save_tModel entity message might have multiple tModel entities, if any one of the tModel entities is a categorization tModel entity and it fails validation, the request fails with a UDDIInvalidValueException plus additional information to explain the cause, and none of the tModel entities is published. For example:

```
E_invalidValue (20200) A value that was passed in a keyValue attribute did not
pass validation.  This applies to checked categorizations, identifiers and
other validated code lists. The error text will clearly indicate the key and
value combination that failed validation. Invalid 'customTaxonomy:dbKey'
keyValue [naics] in keyedReference. KeyValue already in use by
tModelKey[UUID:C0B9FE13-179F-413D-8A5B-5004DB8E5BB2]
```

UDDI Utility Tools

The UDDI Utility Tools is a suite of functions that you can use to migrate, move, or copy UDDI Version 2 entities, including child entities and their respective Version 2 entity keys, into a Version 3 UDDI registry.

To use the UDDI Utility Tools suite, run the UDDIUtilityTools.jar file. This file is in the *app_server_root*/UDDIReg/scripts directory. Alternatively, you can invoke all the functions of UDDI Utility Tools through the supplied public Java API.

The UDDI Utility Tools suite that is supplied in this release has the same functions as the version that is supplied in WebSphere Application Server Version 6.1. However, all UDDI Utility Tools functions in this release use the UDDI Version 2 API. Note that Version 2 API does not support publisher-assigned keys.

You can use the UDDI Utility Tools to export from Version 2 and 3 registries (supplying only the Version 2 representation of the UDDI entity key) and import into the Version 3 registry, using Version 2 API types.

Entities from a Version 3 registry are exported as Version 2 entities, so elements such as digital signatures are not present. See the topic about saving UDDI Version 3 entities with a supplied key for an example of how to use the Version 3 API to assign your own keys to Version 3 entities.

The UDDI Version 3 Publish API supports publisher-assigned keys, and to promote entities between Version 3 registries, you use normal API functions.

The UDDI Utility Tools suite also has the following uses:

- You can search and select entities from a source UDDI registry by specifying Version 2 keys or search criteria.
- You can publish canonical tModel entities in a UDDI registry, including child entities.
- You can persist UDDI (Version 2) entities in an intermediate XML representation so that you can customize and copy those entities to multiple target UDDI registries, by specifying Version 2 keys.
- You can update existing entities in a target UDDI registry, including child entities.
- You can delete selected entities from a target UDDI registry by specifying Version 2 keys.

The UDDI Utility Tools suite has five main functions:

Export
: The export function gets a list of UDDI entities from a specified registry and writes them to the UDDI entity definition file, using the specified entity types and keys. The entity type for each key can be business, service, bindingTemplate, or tModel. The entity definition file contains XML that exactly describes each of the specified entities, according to the UDDI Utility Tools schema, which includes the UDDI Version 2 schema. The UDDI entity definition file separates entities by type, and automatically detects and records tModel entities that the specified entities reference. You can use the “referenced tModels” section of the file to ensure that a target registry includes any referenced tModel entities before you import new entities to that registry.

Import
: The import function detects whether a list of UDDI entities already exist in the target registry and, if they do not, create a minimal entity, or stub, with the specified key. The UDDI entities can be supplied through a UDDI entity definition file, or programmatically in a container object. The entities are then published, updating the stubs with the supplied data, and overwriting or ignoring existing entities, as specified. Note that the original key is maintained throughout.

Promote
: The promote function combines the export and import steps such that the specified entities are extracted (by key) from the source registry and then imported into the target registry in a single logical step. Optionally, you can generate a UDDI entity definition file.

Delete The delete function deletes the specified entities from the target UDDI registry. The entities to delete are specified as an entity type, or a list of entity types, and keys, in the same way as for the export function.

Find matching entities
: The find matching entities function finds a set of entities that match the search criteria and generates a list of entity keys. The search criteria are UDDI Inquiry API objects for each of the various entity types. You can use the resulting list of entity keys as input to the export, promote, and delete functions.

  Note: This function is available through only the programmatic API.

  The following diagram shows relationship between the functions, their input and output, and the source and target UDDI registries:

UDDI Utility Tools prerequisites

Before you use the UDDI Utility Tools, ensure that the required .jar files are available.

Ensure that the following .jar files are available to the UDDI Utility Tools. You must specify the locations of the .jar files in the class path in the UDDI Utility Tools properties file:

UDDIUtilityTools.jar
This file is the UDDI Utility Tools .jar file and is in the *app_server_root*/UDDIReg/scripts directory.

com.ibm.uddi.jar
This file contains the UDDI4J classes and is in the *app_server_root*/plugins directory.

j2ee.jar
This file contains some required Java platform for enterprise applications classes, and is in the *app_server_root*/lib directory.

com.ibm.ws.runtime.jar
This file is the Apache SOAP implementation and is in the *app_server_root*/plugins directory.

DbDriver
This driver is needed for the UDDIUtilityTool to connect to your target database. See the following table for the values you must specify for your chosen database:

Table 337. DbDriver values for databases. The table details the values needed to connect to different databases.

	DB2	**Apache Derby**	**Oracle**
DBDriverLocation for class path	*DB2_HOME*/db2java.zip	*app_server_root*/derby/lib/derbyclient.jar	*ORACLE_HOME*/jdbc/lib/ojdbc6.jar
Driver	com.ibm.db2.jdbc.app.DB2Driver, or com.ibm.db2.jcc.DB2Driver for a remote DB2 database. You can also set up a local alias to the remote database by using the DB2 client.	com.ibm.db2.jcc.DB2Driver	oracle.jdbc.OracleDriver
URL	jdbc:db2://host:*database_name*	jdbc:db2j:net://host:1527/*database_name*	jdbc:oracle:thin:@host:1521:*database_name*

where:

- *app_server_root* is the directory location of WebSphere Application Server.
- *DB2_HOME* is the directory location of DB2, for example `c:\Program Files\SQLLIB\java12\`
- *ORACLE_HOME* is the directory location of Oracle, for example `c:\oracle\ora92\`
- *database_name* is the name of the database. For Apache Derby, ensure that *database_name* includes the path to the database, for example *profile_root*/databases/com.ibm.uddi/UDDI30

Notes:

- For Apache Derby, make the database network-enabled so that it can handle multiple connections. For further details, refer to the section about managing the Derby Network Server in the Derby Server and Administration Guide.
- For DB2, add DB2_HOME/sqllib/lib to your LD_LIBRARY_PATH and LIBPATH environment variables.

The Security provider configuration section in the configuration properties file shows the location of the default DummyClientTrustFile.jks file. If you use your own truststore, ensure that the location is placed here.

The UDDI Utility Tools use UDDI Version 2 SOAP Inquiry and Publish interfaces. These APIs are protected, as described in the topic about access control for UDDI registry interfaces. The UDDI Utility Tools also access the UDDI registry database through the database driver, and access to the database is controlled by the database management system.

UDDI Utility Tools configuration file

Configuration data for UDDI Utility Tools is in a configuration properties file, which describes the runtime environment, UDDI and database locations and access information, logging information, security configuration, entity definition file location, and other flags to control whether referenced entities can be imported, overwritten, or both.

A sample configuration properties file, UDDIUtilityTools.properties, is supplied with UDDI Utility Tools. By default, this file is in the *app_server_root*/UDDIReg/scripts directory. If you do not specify a properties path, by default, the configuration properties file is searched for in the current directory.

To set up and use the configuration file, use the following procedure:

1. Modify the sample configuration properties file:
   - Set the class path, which must include the current directory (.), the UDDIUtilityTools.jar, and all the dependent jars, as listed in "UDDI Utility Tools prerequisites" on page 3556. The class path must include the database driver JAR file, for example, db2java.zip.
   - If you are configuring a Java Secure Socket Extension (JSSE) provider, add the .jar file that contains the provider to the class path. To configure a JSSE provider, set the `jsse.provider` property. The default value is com.ibm.jsse.IBMJSSEProvider. To specify the Federal Information Processing Standard (FIPS) JSSE provider, set the `jsse.provider` property to com.ibm.fips.jsse.IBMJSSEFIPSProvider.
   - Set other properties as required. For details, see the comments in the sample UDDIUtilityTools.properties file.
   - Change localhost to the name of your server.
   - Change the port number 9080 to your internal HTTP port.
2. When you run UDDI Utility Tools, specify the modified configuration properties file.

The following example shows the sample configuration properties file.

```
############################################
# Runtime environment                      #
# (if invoking using java -jar...)         #
# "X Y" required around paths with spaces.  #
# Replace WAS_HOME with your WebSphere      #
# Application Server home path.            #
# db2java.zip is for DB2 - replace this with #
# appropriate database driver file.        #
############################################
classpath=.:WAS_HOME/UDDIReg/scripts/UDDIUtilityTools.jar:
WAS_HOME/plugins/com.ibm.ws.runtime.jar:WAS_HOME/plugins/com.ibm.uddi.jar:
WAS_HOME/dev/javaEE/j2ee.jar:/QIBM/UserData/Java400/ext/db2_classes.jar

############################################
# SOAP entry points for source UDDI        #
############################################
fromInquiryURL=http://localhost:9080/uddisoap/inquiryapi
fromGetURL=http://localhost:9080/uddisoap/get

############################################
# SOAP entry points for target UDDI        #
############################################
toInquiryURL=http://localhost:9080/uddisoap/inquiryapi
toPublishURL=http://localhost:9080/uddisoap/publishapi

############################################
# UDDI registry user information           #
#                                          #
# Note: This information must match the user #
# information that was used to publish the  #
# entities on the target UDDI registry.    #
############################################
userID=UNAUTHENTICATED
password=NONE

############################################
# Configuration for destination UDDI DB    #
# Userid and Password must have authority to #
```

```
# the iSeries server and DB                  #
############################################
dbDriver=com.ibm.db2.jdbc.app.DB2Driver
dbUrl=jdbc:db2:localhost/ibmudi30
dbUser=iSeriesUserProfile
dbPasswd=iSeriesUserPassword

############################################
# Security provider configuration           #
############################################
# Indicates whether security is required on the target registry
secure.connection=true

# The location of the truststore if security is required
trustStore.fileName=TrustFile.jks

# The password for the trust store
trustStore.password=WebAS

# The JSSE Provider class name
jsse.provider=com.ibm.jsse.IBMJSSEProvider

############################################
# Trace and message logging configuration    #
############################################
# detail level of message output (all functions)
verbose=true

# detail level of trace output.
# 1: severe
# 2: normal
# 3: detail
traceLevel=3

# path to message log file (relative or absolute)
messageLogFileName=logs/messages.log

# path to trace log file (relative or absolute)
traceLogFileName=logs/trace.log

############################################
# Miscellaneous Options                      #
############################################
# indicates whether existing entities are overwritten (import/promote)
# Note: tModels in referencedTModels section are never overwritten,
#       regardless of this setting. To overwrite tModels, they must
#       be present in the tModels section.
overwrite=false

# indicates whether referenced entities are imported (import/promote)
importReferencedEntities=true

# location of entity definition file, used for (export/import)
UddiEntityDefinitionFile=definitions/entities01.xml

# namespace prefix to use in definition file (export)
namespacePrefix=promote
```

UDDI entity definition file

The entity definition file contains XML that exactly describes each of the specified entities, according to the UDDI Utility Tools schema.

You can create a UDDI entity definition file in three ways:

- Use the export or promote functions in UDDI Utility Tools to generate a file.
- Modify a file that was generated by using the export function.
- Create a file manually.

The extension to the uddi:tModel type to add a deleted attribute is not currently used in UDDI Utility Tools.

The file is validated for form and compliance with the UDDI Utility Tools schema, shown here:

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema id="uddiPromote" attributeFormDefault="unqualified"
 elementFormDefault="qualified" targetNamespace=
 "http://www.ibm.com/xmlns/prod/WebSphere/UDDIUtilityTools"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:uddi="urn:uddi-org:api_v2"
xmlns="http://www.ibm.com/xmlns/prod/WebSphere/UDDIUtilityTools"
xmlns:promote="http://www.ibm.com/xmlns/prod/WebSphere/UDDIUtilityTools">

 <xsd:import namespace="http://www.w3.org/XML/1998/namespace"
   schemaLocation="xml.xsd" />
 <xsd:import namespace="urn:uddi-org:api_v2" schemaLocation="uddi_v2.xsd" />

 <!-- define a type to represent the state of a tModel -->
 <xsd:simpleType name="tModelDeleted">
  <xsd:restriction base="xsd:NMTOKEN">
   <xsd:enumeration value="true" />
   <xsd:enumeration value="false" />
  </xsd:restriction>
 </xsd:simpleType>

 <!-- extend tModel with additional attribute of type tModelDeleted -->
 <!-- This is restricted to values true or false -->
 <xsd:complexType name="tModel">
  <xsd:complexContent>
   <xsd:extension base="uddi:tModel">
    <xsd:attribute name="deleted" type="promote:tModelDeleted"
          use="optional" />
   </xsd:extension>
  </xsd:complexContent>
 </xsd:complexType>


 <!-- Top level element definitions -->
 <xsd:element name="uddiEntities" type="promote:uddiEntities" />
 <xsd:complexType name="uddiEntities">
  <xsd:sequence>
   <xsd:element ref="promote:tModels" minOccurs="0" maxOccurs="1" />
   <xsd:element ref="promote:businesses" minOccurs="0" maxOccurs="1" />
   <xsd:element ref="promote:services" minOccurs="0" maxOccurs="1" />
   <xsd:element ref="promote:bindings" minOccurs="0" maxOccurs="1" />
   <xsd:element ref="promote:referencedTModels" minOccurs="0"
        maxOccurs="1" />
  </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="businesses" type="promote:businesses" />
 <xsd:complexType name="businesses">
  <xsd:sequence>
   <xsd:element ref="uddi:businessEntity" minOccurs="0"
        maxOccurs="unbounded" />
  </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="tModels" type="promote:tModels" />
 <xsd:complexType name="tModels">
  <xsd:sequence>
   <xsd:element ref="uddi:tModel" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="services" type="promote:services" />
 <xsd:complexType name="services">
  <xsd:sequence>
   <xsd:element ref="uddi:businessService" minOccurs="0"
        maxOccurs="unbounded" />
  </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="bindings" type="promote:bindings" />
 <xsd:complexType name="bindings">
  <xsd:sequence>
   <xsd:element ref="uddi:bindingTemplate" minOccurs="0"
        maxOccurs="unbounded" />
  </xsd:sequence>
 </xsd:complexType>

 <xsd:element name="referencedTModels" type="promote:referencedTModels" />
 <xsd:complexType name="referencedTModels">
```

```
  <xsd:sequence>
   <xsd:element ref="uddi:tModel" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
 </xsd:complexType>
</xsd:schema>
```

UDDI entity definition file example for canonical tModel entities

You can use UDDI Utility Tools to create new UDDI entities in a target UDDI registry. A typical example of this is to introduce a new canonical tModel entity that has a publicly known tModel key.

The following example entity definition file following shows the five main sections for tModels, businesses, services, bindings, and referencedTModels:

```
<?xml version="1.0" encoding="UTF-8"?>
<promote:uddiEntities xmlns="urn:uddi-org:api_v2" xmlns:promote=
 "http://www.ibm.com/xmlns/prod/WebSphere/UDDIUtilityTools">

  <!-- tModels -->
  <promote:tModels>

    <tModel tModelKey="uuid:ee3966a8-faa5-416e-9772-128554343571" >
      <name>http://schemas.xmlsoap.org/ws/2002/07/policytmodel</name>
      <description>WS-PolicyAttachment policy expression</description>
    </tModel>

    <tModel tModelKey="uuid:ad61de98-4db8-31b2-a299-a2373dc97212" >
      <name>uddi-org:wsdl:address</name>
 <description xml:lang="en">
This tModel is used to specify the URL fact that the address must be obtained
from the WSDL deployment file.
      </description>
      <overviewDoc>
        <overviewURL>
http://www.oasis-open.org/committees/uddi-spec/doc/tn/
uddi-spec-tc-tn-wsdl-v2.htm#Address
        </overviewURL>
      </overviewDoc>
    </tModel>

  </promote:tModels>

  <!-- businesses -->
  <promote:businesses>
  </promote:businesses>

  <!-- services -->
  <promote:services>
  </promote:services>

  <!-- bindings -->
  <promote:bindings>
  </promote:bindings>

  <!-- referenced tModels -->
  <promote:referencedTModels>
  </promote:referencedTModels>

</promote:uddiEntities>
```

UDDI Utility Tools at a command prompt

You can start UDDI Utility Tools at a command prompt. In some situations, there are prerequisites before you run the command.

Ensure that you are using the correct level of Java code by setting the PATH statement to include the Java code that is supplied with WebSphere Application Server. For example, from the command line, type:

Use one of the following approaches to start UDDI Utility Tools:

- Enter the following command and use a specified properties file that sets up class path and other parameters:

```
java - jar UDDIUtilityTools.jar {function} [options]
```

Note: Before you run UDDIUtilityTools.jar from the command line, ensure that you edit the UDDIUtilityTools.properties file. If you save this properties file in a different directory from the UDDIUtilityTools.jar file, specify the location of the properties file as part of the command line arguments.

- Enter the following command, where CommandLineProcessor is the class that processes command line arguments for UDDI Utility Tools, sets up the configuration and invokes the appropriate function:

```
java CommandLineProcessor
```

Usage of UDDIUtilityTools.jar:

```
java -jar UDDIUtilityTools.jar {function} [options]
```

Functions:

```
  -promote <entity source>   Promote entities between registries
  -export <entity source>    Extract entities from a registry to XML
  -delete <entity source>    Delete entities from a registry
  -import                    Create entities from XML to a registry

where <entity source> is one of:
  -tmodel|-business|-service|-binding <key> Specify a single entity type and key
  -keysFile | -f <filename>  Specify a file that contains entity types and keys
```

Options:

```
  -properties <filename>     Specify the path to a configuration file
  -overwrite | -o            Overwrite an entity if it already exists
  -log | -v                  Output verbose messages
  -definitionFile <filename> Specify the path to a UDDI entity definition file
  -importReferenced          Import entities that are referenced by source entities
```

The following command-line options override property settings in the configuration file:

- overwrite
- log
- definitionFile
- importReferenced

Examples

```
java -jar UDDIUtilityTools.jar -promote -keysFile /uddikeys.txt
```

Export a single business to the entity definition file that is specified in a properties file in the current directory.

```
java -jar UDDIUtilityTools.jar -export
-business 28B8B928-2B2E-4EC9-A647-1E40651E4752
```

Export a single business to the entity definition file that is specified in a properties file in the current directory and use a keys file to specify the entities to export.

```
java -jar UDDIUtilityTools.jar -export -keysFile /myKeyFiles/keyFile01.txt
```

Export a single business to the entity definition file that is specified in a properties file in the current directory and use a keys file to specify the entities to export. Also, display verbose output on the command line.

```
java -jar UDDIUtilityTools.jar -export -keysFile /myKeyFiles/keyFile02.txt -v
```

Import the contents of the default entity definition file that is specified in a UDDIUtilityTools.properties file in the current directory.

```
java -jar UDDIUtilityTools.jar -import
```

Import the contents of the default entity definition file that is specified in a UDDIUtilityTools.properties file in the current directory and import referenced tModel entities into the target registry.

```
java -jar UDDIUtilityTools.jar -import -importReferenced
```

Import the entities from an entity definition file at the specified location.

```
java -jar UDDIUtilityTools.jar -import -definitionFile /myEDFs/entities01.xml
```

Import the entities from the default entity definition file including referenced tModel entities. The overwrite options specifies that any entities, excluding referenced tModel entities that are found in the target registry, are overwritten.

```
java -jar UDDIUtilityTools.jar -import -overwrite -importReferenced
```

Promote a single service from a source to a target registry using the properties file at a specified location.

```
java -jar UDDIUtilityTools.jar -promote
-service 67961D67-330F-4F14-8210-E74A58E710F3
-properties /UUT/myUUTProps.properties
```

Promote a set of entities that is specified in a keys file.

```
java -jar UDDIUtilityTools.jar -promote -keysFile /myKeyFiles/keyFile03.txt
```

Promote a set of entities that is specified in a keys file and overwrite existing entities in the target registry.

```
java -jar UDDIUtilityTools.jar -promote -keysFile /myKeyFiles/keyFile04.txt
-overwrite
```

Promote a set of entities that is specified in a keys file, including referenced tModel entities.

```
java -jar UDDIUtilityTools.jar -promote -keysFile /myKeyFiles/keyFile05.txt
-importReferenced
```

Promote a set of entities that is specified in a keys file, but also create an entity definition file that contains the promoted entities.

```
java -jar UDDIUtilityTools.jar -promote -keysFile /myKeyFiles/keyFile06.txt
-definitionFile /myEDFs/entities02.xml
```

Logically delete a single tModel entity. You cannot physically delete tModel entities.

```
java -jar UDDIUtilityTools.jar -delete
-tModel UUID:1E2B9D1E-E53D-4D36-9D46-6CCC176C466A
```

Delete all the entities that are specified in the keys file. Except for tModel entities, all other entities are physically deleted from the target registry.

```
java -jar UDDIUtilityTools.jar -delete -keysFile /myKeyFiles/keyFile04.txt
```

A keys file example

The following example shows the keys to export, promote, or delete from the target registry:

```
#
# Keys of entities to be exported, promoted from source registry or deleted
# from target registry
#
# Note: keys must be comma separated and on SAME line
# Note: property names are case sensitive. ('tmodels=' are ignored)

businesses=97C77097-AC6C-4CA0-A6C4-452F7045C470,
4975E949-581F-4FCA-AD5F-E08280E05F9F
services=BB3864BB-1578-4833-8179-14391F14791F
bindings=
tModels=273F1727-7BFF-4FB5-A1FD-BA5C45BAFD9C
```

If the importReferenced property is set to `true`, the list of tModel entities in the referencedTModels section is imported to the target registry. If the referencedTModel is new, minimal entities are created. If the referencedTModel already exists, it is never overwritten, regardless of the overwrite property value. This approach prevents commonly referenced tModel entities, such as categorization tModel entities, from being updated unnecessarily.

If you want to update a referencedTModel, you must manually move the referencedTModel definition to the tModel entities section in the entity definition file and set overwrite to `true`.

UDDI Utility Tools log files:

The following examples show the contents of two log files that are produced by running UDDI Utility Tools. The examples include some comments in square brackets and in italics to highlight important points in the log file.

The following file is the messages.log file, which shows successful and unsuccessful operations for export, import, and delete functions:

```
[29/07/04 17:39:57:531 BST] CWUDU0002I: ******Starting UDDI Utility Tools *******
[timestamp and eyecatcher indicate when tool is run]
[29/07/04 17:39:57:531 BST] CWUDU0009I: Exporting entities...
[29/07/04 17:39:57:531 BST] CWUDU0015I: Exported 14 entities.
[29/07/04 17:39:57:531 BST] CWUDU0029I: Serializing...
[29/07/04 17:39:57:531 BST] CWUDU0030I: Serialized entities.
[29/07/04 17:39:57:531 BST] CWUDU0016I: Importing entities...
[29/07/04 17:39:57:531 BST] CWUDU0124I: Created tModel minimal entity with
tModelKey [uuid:667e2766-4781-4151-b3a0-809f7180a096].
[29/07/04 17:39:57:531 BST] CWUDU0121I: Created business minimal entity with
businessKey [263f5526-8708-4834-9f5d-8f8c878f5d6e].
[29/07/04 17:39:57:531 BST] CWUDU0122I: Created service minimal entity with
serviceKey [0af2a30a-be70-401f-a027-331a6c332712].
[29/07/04 17:39:57:531 BST] CWUDU0122I: Created service minimal entity with
serviceKey [61012761-d02c-4c70-ae98-435ffd4398f9].
[29/07/04 17:39:57:531 BST] CWUDU0123I: Created binding template minimal
entity with bindingKey [f97af9f9-7cb7-47bd-8b90-b55e4db590df].
[29/07/04 17:39:57:531 BST] CWUDU0123I: Created binding template minimal
entity with bindingKey [17e4c017-d273-43ec-af4a-f9b841f94a30].
[29/07/04 17:39:57:531 BST] CWUDU0123I: Created binding template minimal
entity with bindingKey [9e2c239e-3b30-40a9-9c25-ce64edce25b9].
[29/07/04 17:39:57:531 BST] CWUDU0121I: Created business minimal entity with
businessKey [49bb6949-4b0e-4e81-88a7-e26bfbe2a7f1].
[29/07/04 17:39:57:531 BST] CWUDU0122I: Created service minimal entity with
serviceKey [003d2b00-f6c0-4071-8b84-f235a2f28445].
[29/07/04 17:39:57:531 BST] CWUDU0123I: Created binding template minimal
entity with bindingKey [df1019df-2d2f-4f32-bf18-4f21274f1835].
[29/07/04 17:39:57:531 BST] CWUDU0123I: Created binding template minimal
entity with bindingKey [b229aeb2-f2b1-4115-a06f-536753536f10].
[29/07/04 17:39:57:531 BST] CWUDU0122I: Created service minimal entity with
serviceKey [84d8e584-2510-4099-9b2a-6023f1602a0a].
[29/07/04 17:39:57:531 BST] CWUDU0123I: Created binding template minimal
entity with bindingKey [62a9a762-7fff-4f7a-8463-af0c79af63ee].
[29/07/04 17:39:57:531 BST] CWUDU0123I: Created binding template minimal
entity with bindingKey [e08654e0-b212-42c0-bcf3-655e9765f392].
[29/07/04 17:39:57:531 BST] CWUDU0115I: Imported 7 entities and 0 referenced
entities.
[this kind of message indicates that the operation worked!]
[29/07/04 17:39:57:531 BST] CWUDU0002I: ****** Starting UDDI Utility Tools ******
[29/07/04 17:39:57:531 BST] CWUDU0023I: Deleting entities...
[29/07/04 17:39:57:531 BST] CWUDU0028I: Deleted 7 entities.
```

Note: This topic references one or more of the application server log files. Beginning in WebSphere Application Server Version 8.0 you can configure the server to use the High Performance Extensible Logging (HPEL) log and trace infrastructure instead of using `SystemOut.log` , `SystemErr.log`, `trace.log`, and `activity.log` files or native z/OS logging facilities. If you are using HPEL, you can access all of your log and trace information using the LogViewer command-line tool from your server profile bin directory. See the information about using HPEL to troubleshoot applications for more information on using HPEL.

The following log file shows a typical trace log file entry for an export:

```
[29/07/04 17:39:57:531 BST] ********** Starting UDDI Utility Tools **********
[eyecatcher and timestamp indicate when tool is run]
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.PromoterAPI.setUddiEntities()
[the '>' indicates entry to the constructor of this class]
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.export.KeyFileReader()
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.export.KeyFileReader()
loaded tModel keys
```

```
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.export.KeyFileReader()
loaded business keys
TransformConfiguration:
  nameSpacePrefix=promote
  uddiEntityDefinitionFile=/temp/MigToolFiles/Results/Promote_api_EDF_1.xml

ExportConfiguration:
  fromGetURL=http://yottskry:9080/uddisoap/
  fromInquiryURL=http://yottskry:9080/uddisoap/inquiryAPI

ImportConfiguration:
  overwrite=true
  uddiEntityDefinitionFile=/temp/MigToolFiles/Results/Promote_api_EDF_1.xml
  importReferencedEntities=true

PublishConfiguration:
  toInquiryURL=http://davep:9080/uddisoap/inquiryAPI
  toPublishURL=http://yottskry:9080/uddisoap/publishAPI
  userID=Publisher1
  trustStoreFileName=/WebSphere600/AppServer/etc/DummyClientTrustFile.jks
  secureConnection=false

DatabaseConfiguration:
  dbDriver=com.ibm.db2.jcc.DB2Driver
  dbURL=jdbc:db2:LOC1
  dbUser=db2admin

LoggerConfiguration:
  messageStream=null
  messageLogFileName=/temp/MigToolFiles/logs/message.log
  traceLogFileName=/temp/MigToolFiles/logs/trace.log
  traceLevel=3
  verbose=true


[29/07/04 17:39:57:531 BST] < com.ibm.uddi.promoter.PromoterAPI()
[29/07/04 17:39:57:531 BST] ********** Starting UDDI Utility Tools **********
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.PromoterAPI.setUddiEntities()
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.export.KeyFileReader()
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.export.KeyFileReader()
loaded tModel keys
[ log entries without a '>' or '<' are status messages only ]
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.export.KeyFileReader()
loaded business keys
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.export.KeyFileReader()
loaded service keys
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.export.KeyFileReader()
loaded binding keys
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.UddiEntityKeys()
[29/07/04 17:39:57:531 BST] < com.ibm.uddi.promoter.UddiEntityKeys()
[the '<' indicates exit from the constructor]
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.export.KeyFileReader()
removed duplicate, empty and null keys
[29/07/04 17:39:57:531 BST] < com.ibm.uddi.promoter.export.KeyFileReader()
[29/07/04 17:39:57:531 BST] < com.ibm.uddi.promoter.PromoterAPI.setUddiEntities()
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.PromoterAPI.deleteEntities()
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.publish.EntityDeleter()
[29/07/04 17:39:57:531 BST] < com.ibm.uddi.promoter.publish.EntityDeleter()
[29/07/04 17:39:57:531 BST] > com.ibm.uddi.promoter.UDDIClient()
[29/07/04 17:39:57:531 BST]   com.ibm.uddi.promoter.UDDIClient() client type: 1
```

UDDI Utility Tools through the API

UDDI Utility Tools provides a public API to functions to export, import, promote, find, and delete UDDI entities. To invoke these functions, use the PromoterAPI class.

Typically, you use these functions through the PromoterAPI class for the following uses:

- Create a Configuration object and populate it from a Properties object or from a configuration properties file.
- Create a PromoterAPI object, passing the Configuration object in the constructor.
- For keys based functions (export, delete, and promote), set the keys by supplying a UDDIEntityKeys object, the location of the keys file, or, for one entity, by specifying an entity type and a key value.

- Invoke the corresponding method for the function required: exportEntities, promoteEntities(boolean), importEntities, deleteEntities, or extractKeysFromInquiry(FindTModel, FindBusiness, FindService, FindBinding, FindRelatedBusinesses).

The samples for WebSphere Application Server include sample code for UDDI Utility Tools that demonstrates use of the API classes.

Note: The low-level UDDI Utility Tools API classes and methods, such as BusinessStub and ServiceStub, are deprecated in WebSphere Application Server Version 6.0. These APIs are replaced with the high-level PromoterAPI interface in the com.ibm.uddi.promoter package. Refer to the API documentation for details.

Save UDDI Version 3 entities with a supplied key

The following example shows how to save a Version 3 business entity with a defined key. You might do this after you use UDDI Utility Tools to export from a Version 3 registry, because entities from a Version 3 registry are exported as Version 2 entities, so elements such as digital signatures are not present.

```
<?xml version="1.0" encoding="UTF-8"?>
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <save_business xmlns="urn:uddi-org:api_v3">
      <authInfo>a399c4a3-6387-47cd-a1bd-91f7bb91bdd7</authInfo>
      <businessEntity businessKey="uddi:mycompany-p1.com:computers">
        <name xml:lang="en">WithKey</name>
      </businessEntity>
    </save_business>
  </Body>
</Envelope>
```

UDDI Utility Tools limitations and resolutions

For some limitations with the UDDI Utility Tools, there are actions to resolve each issue.

- You attempt to run UDDI Utility Tools, but you get the following error:

```
java.lang.noClassDefFoundError: com/ibm/uddi/promoter/CommandLineProcessor
```

  - Ensure that you have edited the supplied properties file, `UDDIUtilityTools.properties` and that the file contains appropriate values for your environment. For details of the files that are required on the classpath, and an example properties file, see the topics about UDDI Utility Tools prerequisites and the UDDI Utility Tools configuration file.
  - Ensure that you use the level of Java code that is supplied with WebSphere Application Server.
- PublisherAssertions are not supported and are not promoted.

  To resolve this issue, after you promote the businesses that are related, recreate the publisherAssertion relationship.
- Referenced businesses in service projections are not added automatically to the entity definition file in the same manner as referenced tModel entities.

  To resolve this issue, add the referenced business that owns the projected service to the entity definition file. If the business is not present in the target registry, place it before the owning business of the service in the entity definition file.
- Cycle detection for service projections are not detected in the same manner as for referenced tModel entities.

  To resolve this issue, if a circular reference is present between two or more service projections, break the cycle by removing one of the projections temporarily, perform the import, and update the changed entity to establish the cycle again in the target registry.
- The tModel entities that are deleted, in the logical sense, in the source registry are imported and promoted as undeleted in the target registry. This is because, in the UDDI Version 2 specification, the deleted state of tModel entities is not exposed as API calls.

  To resolve this issue, after importing the tModel entity, perform a delete, using the UDDI Utility Tools delete function, or any other UDDI registry API access method.

- BindingTemplates that are referenced by hostingRedirectors are not added automatically to the entity definition file in the same manner as referenced tModel entities.

  To resolve this issue, add the referenced bindingTemplate to the entity definition file.
- Businesses that are referenced by an owningBusiness keyedReference are not added automatically to the entity definition file.

  To resolve this issue, import the referenced business into the target registry before importing the tModel that references it.
- With an embedded Apache Derby database, the import and promote functions are not supported when referencing a target registry.

  To resolve this issue, make the embedded Apache Derby database network-enabled. For information about configuring network Apache Derby, refer to the section about managing the Derby Network Server in the Derby Server and Administration Guide.

  The export and delete functions when referencing a source registry with an embedded Apache Derby database are supported.
- Some combinations of command-line arguments are not validated and prevented. For example, it is possible to specify `-import` with `-keysFile` *`path_to_file`* in the same command, although the `-keysFile` option is ignored.

Chapter 40. Administering Work area

This page provides a starting point for finding information about work areas, a WebSphere extension for improving developer productivity.

Work areas provide a capability much like that of global variables. They enable efficient sharing of information across a distributed application.

For example, you might want to add profile information as each customer enters your application. By placing this information in a work area, it is available throughout your application, eliminating the need to hand-code a solution or to read and write information to a database.

Managing the UserWorkArea partition

Managing the UserWorkArea partition

Before you begin

For an application to take advantage of work areas, the work area service must be enabled for both clients and servers. On a server the service is disabled by default. On the client, the service is enabled by default.

For an application to take advantage of the default partition, the UserWorkArea partition, this partition must be enabled by enabling the work area service for both clients and servers. The work area service on a server is disabled by default and the work area service on a client is enabled by default. Note that rather than using this default work area partition, a user can create their own work area partition using the Work area partition service.

About this task

Applications can set maximum sizes on each work area that is sent or received. By default, the maximum size of a work area that is sent by a client and received, then possibly resent, by a server is 32,768 bytes. The maximum size that you can specify is determined by the maximum value expressible in the Java Integer data type, 2,147,483,647. The smallest maximum size that you can specify is 1. Using a maximum size of 1 byte effectively means that no requests associated with the work area can leave the system or enter another system. A value of 0 means that no limit is imposed. A value of -1 means that the default value is to be honored. The default value is also used if an invalid value or a malformed property is specified. You can change this size as described in this topic.

Procedure

1. Enable or disable the use of the UserWorkArea partition on a server: The work area service is disabled by default on servers but enabled by default on the client
   a. Start the administrative console.
   b. Select **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Business Process Services** > **Work area service**.
   c. Select or clear the **Startup** check box. This specifies whether or not the server should automatically start the work area service when the server starts.
   d. Save the new configuration and restart the server to apply the new configuration.
2. Enable (or disable) the UserWorkArea partition on a client: Set the com.ibm.websphere.workarea.enabled property to TRUE or FALSE before starting the client. For example, to disable the work area service, when invoking the launchClient script found in the *app_server_root*/`bin` directory, add the following system property to the launchClient invocation:

   ```
   -CCDcom.ibm.websphere.workarea.enabled=false
   ```

Alternatively, this property can be set in a property file that is used by the launchClient script. Refer to the Running a Java EE client application with launchClient article for additional information.

3. Manage the size of the work areas that this server can send and the number of work areas that this server can accept.
   a. Start the administrative console.
   b. Select **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Business Process Services** > **Web container**.
      - To change the send size or receive size on the work area service (namely the "UserWorkArea" partition):
        – Select **Work area service**.
      - To change the send size or receive size on a user defined partition:
        – Select **Work area partition service**.
        – Select a partition.
   c. Enter a new value in the **Maximum send size** field to modify the size of the work area that this server can send, or enter a new value in the **Maximum receive size** field to modify the size of the work area that this server can accept.
   d. Save the new configuration and restart the server to apply the new configuration.
4. Change the size of the work area that can be sent by a client. This step only applies to the UserWorkArea partition on the client. To set the maximum send or receive size on a user defined partition, you must set these values when creating the partition on the client. For more information on creating a partition on a client, see the client section in the Configuring work area partitions topic. To change the size of the work area that can be sent by a client, set the com.ibm.websphere.workarea.maxSendSize property to the desired number of bytes before starting the client. You can set the maximum send size as follows:
   - Set the maximum send size when invoking the launchClient invocation script found in the `$WAS_HOME/bin` directory. For example, to set the maximum size to 10,000 bytes, add the following system properties to the launchClient invocation as needed:
     `-CCDcom.ibm.websphere.workarea.maxSendSize=10000`
   - Set the maximum send size propery, com.ibm.websphere.workarea.maxSendSize, in a property file that is used by the launchClient script. Refer to the Running a Java EE client application with launchClient article for additional information.

   Because the UserWorkArea partition is defined as unidirectional, for example, context only propagates on outbound calls and not on the return of those calls, the maximum receive size is ignored.

Accessing the UserWorkArea partition

About this task

The work area service provides a JNDI binding to an implementation of the UserWorkArea interface under the name `java:comp/websphere/UserWorkArea`. This is the default work area partition, namly the "UserWorkArea" partition. It is created and bound into JNDI naming automatically, as long as it is enabled as defined in Enabling the work area service (UserWorkArea partition). Applications that need to access UserWorkArea partition can perform a lookup on that JNDI name, as shown in the following code example:

Example

```
import com.ibm.websphere.workarea.*;
import javax.naming.*;

public class SimpleSampleServlet {
  ...

  InitialContext jndi = null;
  UserWorkArea userWorkArea = null;
  try {
```

```
    jndi = new InitialContext();
    userWorkArea = (UserWorkArea)jndi.lookup(
       "java:comp/websphere/UserWorkArea");
  }
  catch (NamingException e) { ... }
}
```

Rather than using this default work area partition, a user has the option to create their own work area partition using the Work area partition service.

What to do next

The next step is to use the begin method to create a new work area and associate it with the calling thread, as described in the Beginning a new work area article.

Managing local work with a work area

Managing local work with a work area

Before you begin

Be sure that your client has a reference to the UserWorkArea interface, as described in the Accessing the UserWorkArea partition topic, or a reference to a user defined partition, as defined in the Accessing a user defined work area partition topic. The following steps use the UserWorkArea partition as an illustration. However a user defined partition can be used in the exact same way.

About this task

In a business application that uses work areas, server objects typically retrieve the work area properties and use them to guide local work.

Procedure

1. Retrieve the name of the active work area to determine whether the calling thread is associated with a work area.

   Applications use the getName method on the UserWorkArea interface to retrieve the name of the current work area. If the thread is not associated with a work area, the getName method returns null. In the following code example, the name of the work area corresponds to the name of the class in which the work area was begun.

   ```
   public class SimpleSampleBeanImpl implements SessionBean {

       ...

       public String [] test() {
          // Get the work-area reference from JNDI.
          ...

          // Retrieve the name of the work area. In this example,
          // the name is used to identify the class in which the
          // work area was begun.
          String invoker = userWorkArea.getName();
          ...
       }
   }
   ```

2. Overriding work area properties. Server objects can override client work area properties by creating their own, nested work area. Refer to the Overriding work area properties article for more information.
3. Retrieve properties from a work area by using the get method.

   The get method is intentionally lightweight; there are no declared exceptions to handle. If there is no active work area, or if there is no such property set in the current work area, the get method returns null.

Important: The get method can raise a NotSerializableError in the relatively rare scenario in which CORBA clients set composed data types and invoke enterprise-bean interfaces.

The following example shows the retrieval of the site-identifier and priority properties by the SimpleSampleBean. Notice that one property was set into an outer work area by the client and the other property was set into the nested work area by the server-side bean; the nesting is transparent to the retrieval of the properties.

```
public class SimpleSampleBeanImpl implements SessionBean {

    public String [] test() {
      ...

      // Begin a nested work area.
      userWorkArea.begin("SimpleSampleBean");
      try {
        userWorkArea.set("company",
                         SimpleSampleCompany.London_Development);
      }
      catch (NotOriginator e) {
      }

      SimpleSampleCompany company =
         (SimpleSampleCompany) userWorkArea.get("company");
      SimpleSamplePriority priority =
         (SimpleSamplePriority) userWorkArea.get("priority");
       ...
   }
}
```

4. Optional: Retrieve a list of all the keys visible from a work area.

   The UserWorkArea interface provides the retrieveAllKeys method for retrieving a list of all the keys visible from a work area. This method takes no arguments and returns an array of strings. The retrieveAllKeys method returns null if there is no work area associated with the thread. If there is an associated work area that does not contain any properties, the method returns an array of size 0.

5. Query the mode of a work area property using the getMode method.

   The UserWorkArea interface provides the getMode method determine the mode of a specific property. This method takes the property's key as an argument and returns the mode as a PropertyModeType object. If the specified key does not exist in the work area, the method returns PropertyModeType.normal, indicating that the property can be set and removed without error.

6. Optional: Delete a work area property.

   The UserWorkArea interface provides the remove method to delete a property from the current scope of a work area. If the property was initially set in the current scope, removing it deletes the property. If the property was initially set in an enclosing work area, removing it deletes the property until the current scope is completed. When the current work area is completed, the deleted property is restored.

   The remove method takes the property's key as an argument. Only properties with the modes normal and read-only can be removed. Attempting to remove a fixed property creates the PropertyFixed exception. Attempting to remove properties in work areas that originated in other processes creates the NotOriginator exception.

Example

The server side of the SimpleSample application example, which is included in the Developing applications that use work areas topic, accepts remote invocations from clients. With each remote call, the server also gets a work area from the client if the client has created one. The work area is propagated transparently. None of the remote methods includes the work area on its argument list.

In the example application, the server objects use the work area interface for demonstration purposes only. For example, the SimpleSampleBean intentionally attempts to write directly to an imported work area, which creates the NotOriginator exception. Likewise, the bean intentionally attempts to mask the read only

SimpleSampleCompany, which triggers the PropertyReadOnly exception. The SimpleSampleBean also nests a work area and successfully overrides the priority property before invoking the SimpleSampleBackendBean. A true business application would extract the work area properties and use them to guide the local work. The SimpleSampleBean mimics this by writing a message that function is denied when a request emanates from a sales environment.

Work area service settings

Use this page to manage the work area service.

The work area service manages the scope and implicit propagation of application context. The work area service panel in the administrative console configures the UserWorkArea partition only and has no effect on the work area partition service panel.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Business Process Services** > **Work area service**.

For additional information about work area, see the com.ibm.websphere.workarea package in the Application Programming Interfaces (API) documentation. The generated API documentation is available in the information center table of contents from the path **Reference** > **>Programming interfaces** > **APIs - Application Programming Interfaces**.

Enable service at server startup:

Specifies whether the server attempts to start the work area service.

Selected
When the application server starts, it attempts to start the work area service automatically.

Cleared
The server does not try to start the work area service. If work areas are used on this application server, the system administrator must start the service manually or select this property and then restart the server.

Maximum send size:

Specifies the maximum size of data that can be sent within a single work area.

Data type	Integer
Units	Bytes
Default	10000
Range	-1, 0 (no limit) and 1 to 2147483647

The following values are also used to define the maximum send size.

-1	Default.
0	No limit.

Maximum receive size:

Specifies the maximum size of data that a single work area can receive.

Data type	Integer
Units	Bytes
Default	10000
Range	-1, 0 (no limit) and 1 to 2147483647

The following values are also used to define the maximum receive size.

-1	Default.
0	No limit.

Enable Web service propagation:

Specifies whether the work area is propagated on Web service requests. This option is disabled by default.

Overriding work area properties

About this task

Work areas are inherently associated with the process that creates them. In the sample application, the client begins a work area and sets into it the site-identifier and priority properties. This work area is propagated to the server when the client makes a remote invocation.

Applications nest work areas in order to temporarily override properties imported from a client process. The nesting mechanism is automatic; invoking begin on the UserWorkArea interface from within the scope of an existing work area creates a nested work area that inherits the properties from the enclosing work area. Properties set into the nested work area are strictly associated with the process in which the work area was begun; the nested work area must be completed within the process that created them. If a work area is not completed by the creating process, the work-area facility terminates the work area when the process exits. After a nested work area is completed, the original view of the enclosing work area is restored. However, the view of the complete set of work areas associated with a thread cannot be decomposed by downstream processes.

Applications set properties into a work area using property modes in ensure that a particular property is fixed (not removable) or read-only (not overrideable) within the scope of the given work area.

Example

In the following code example, the server-side sample bean attempts to write directly to the imported work area; because the UserWorkArea partition is not defined to be bidirectional, this action is not permitted, and the NotOriginator exception is thrown. When the UserWorkArea partition is not defined as bidirectional, the sample bean must begin its own work area in order to override any imported properties, as shown in the second code example. If a work area in a user defined partition is used and is defined as bidirectional, this bean can set context into the work area before beginning another work area. This context set in the bidirectional case propagates back to the caller. Refer to the Work area partition service article for additional information.

```
public class SimpleSampleBeanImpl implements SessionBean {

   public String [] test() {
     ...
     String invoker = userWorkArea.getName();

     try {
       userWorkArea.set("key", "value");
     }
     catch (NotOriginator e) {
     }
     ...
  }
}
```

The following code example demonstrates beginning a nested work area, using the name of the creating class to identify the nested work area.

```
public class SimpleSampleBeanImpl implements SessionBean {

   public String [] test() {
      ...
      String invoker = userWorkArea.getName();
      try {
        userWorkArea.set("key", "value");
      }
      catch (NotOriginator e) {
      }

      // Begin a nested work area. By using the name of the creating
      // class as the name of the work area, we can avoid having
      // to explicitly set the name of the creating class in
      // the work area.
      userWorkArea.begin("SimpleSampleBean");

      ...
   }
}
```

In the example application, the client sets the site-identifier property as read-only; that guarantees that the request is always associated with the client's company identity. A server cannot override that value in a nested work area. In the following code example, the SimpleSampleBean attempts to change the value of the site-identifier property in the nested work area it created.

```
public class SimpleSampleBeanImpl implements SessionBean {

  public String [] test() {
      ...

      String invoker = userWorkArea.getName();
      try {
        userWorkArea.set("key", "value");
      }
      catch (NotOriginator e) {
      }

      // Begin a nested work area.
      userWorkArea.begin("SimpleSampleBean");

      try {
        userWorkArea.set("company",
                         SimpleSampleCompany.London_Development);
      }
      catch (NotOriginator e) {
      }
      ...
  }
}
```

retrieveAllKeys method

About this task

The UserWorkArea interface provides the retrieveAllKeys method for retrieving a list of all the keys visible from a work area. This method takes no arguments and returns an array of strings. The retrieveAllKeys method returns null if there is no work area associated with the thread. If there is an associated work area that does not contain any properties, the method returns an array of size 0.

For additional information about work area, see the com.ibm.websphere.workarea package in the API documentation. The generated API documentation is available in the information center table of contents from the path **Reference** > **APIs - Application Programming Interfaces**.

Managing local work with a work area

Managing local work with a work area

Before you begin

Be sure that your client has a reference to the UserWorkArea interface, as described in the Accessing the UserWorkArea partition topic, or a reference to a user defined partition, as defined in the Accessing a user defined work area partition topic. The following steps use the UserWorkArea partition as an illustration. However a user defined partition can be used in the exact same way.

About this task

In a business application that uses work areas, server objects typically retrieve the work area properties and use them to guide local work.

Procedure

1. Retrieve the name of the active work area to determine whether the calling thread is associated with a work area.

   Applications use the getName method on the UserWorkArea interface to retrieve the name of the current work area. If the thread is not associated with a work area, the getName method returns null. In the following code example, the name of the work area corresponds to the name of the class in which the work area was begun.

   ```
   public class SimpleSampleBeanImpl implements SessionBean {

       ...

       public String [] test() {
          // Get the work-area reference from JNDI.
          ...

          // Retrieve the name of the work area. In this example,
          // the name is used to identify the class in which the
          // work area was begun.
          String invoker = userWorkArea.getName();
          ...
      }
   }
   ```

2. Overriding work area properties. Server objects can override client work area properties by creating their own, nested work area. Refer to the Overriding work area properties article for more information.
3. Retrieve properties from a work area by using the get method.

   The get method is intentionally lightweight; there are no declared exceptions to handle. If there is no active work area, or if there is no such property set in the current work area, the get method returns null.

   Important: The get method can raise a NotSerializableError in the relatively rare scenario in which CORBA clients set composed data types and invoke enterprise-bean interfaces.

   The following example shows the retrieval of the site-identifier and priority properties by the SimpleSampleBean. Notice that one property was set into an outer work area by the client and the other property was set into the nested work area by the server-side bean; the nesting is transparent to the retrieval of the properties.

   ```
   public class SimpleSampleBeanImpl implements SessionBean {

       public String [] test() {
         ...

         // Begin a nested work area.
   ```

```
        userWorkArea.begin("SimpleSampleBean");
        try {
          userWorkArea.set("company",
                           SimpleSampleCompany.London_Development);
        }
        catch (NotOriginator e) {
        }

        SimpleSampleCompany company =
           (SimpleSampleCompany) userWorkArea.get("company");
        SimpleSamplePriority priority =
           (SimpleSamplePriority) userWorkArea.get("priority");
         ...
    }
}
```

4. Optional: Retrieve a list of all the keys visible from a work area.

   The UserWorkArea interface provides the retrieveAllKeys method for retrieving a list of all the keys visible from a work area. This method takes no arguments and returns an array of strings. The retrieveAllKeys method returns null if there is no work area associated with the thread. If there is an associated work area that does not contain any properties, the method returns an array of size 0.

5. Query the mode of a work area property using the getMode method.

   The UserWorkArea interface provides the getMode method determine the mode of a specific property. This method takes the property's key as an argument and returns the mode as a PropertyModeType object. If the specified key does not exist in the work area, the method returns PropertyModeType.normal, indicating that the property can be set and removed without error.

6. Optional: Delete a work area property.

   The UserWorkArea interface provides the remove method to delete a property from the current scope of a work area. If the property was initially set in the current scope, removing it deletes the property. If the property was initially set in an enclosing work area, removing it deletes the property until the current scope is completed. When the current work area is completed, the deleted property is restored.

   The remove method takes the property's key as an argument. Only properties with the modes normal and read-only can be removed. Attempting to remove a fixed property creates the PropertyFixed exception. Attempting to remove properties in work areas that originated in other processes creates the NotOriginator exception.

Example

The server side of the SimpleSample application example, which is included in the Developing applications that use work areas topic, accepts remote invocations from clients. With each remote call, the server also gets a work area from the client if the client has created one. The work area is propagated transparently. None of the remote methods includes the work area on its argument list.

In the example application, the server objects use the work area interface for demonstration purposes only. For example, the SimpleSampleBean intentionally attempts to write directly to an imported work area, which creates the NotOriginator exception. Likewise, the bean intentionally attempts to mask the read only SimpleSampleCompany, which triggers the PropertyReadOnly exception. The SimpleSampleBean also nests a work area and successfully overrides the priority property before invoking the SimpleSampleBackendBean. A true business application would extract the work area properties and use them to guide the local work. The SimpleSampleBean mimics this by writing a message that function is denied when a request emanates from a sales environment.

Work area service settings

Use this page to manage the work area service.

The work area service manages the scope and implicit propagation of application context. The work area service panel in the administrative console configures the UserWorkArea partition only and has no effect on the work area partition service panel.

To view this administrative console page, click **Servers** > **Server Types** > **WebSphere application servers** > ***server_name*** > **Business Process Services** > **Work area service**.

For additional information about work area, see the com.ibm.websphere.workarea package in the Application Programming Interfaces (API) documentation. The generated API documentation is available in the information center table of contents from the path **Reference** > **>Programming interfaces** > **APIs - Application Programming Interfaces**.

Enable service at server startup

Specifies whether the server attempts to start the work area service.

Selected
When the application server starts, it attempts to start the work area service automatically.

Cleared
The server does not try to start the work area service. If work areas are used on this application server, the system administrator must start the service manually or select this property and then restart the server.

Maximum send size

Specifies the maximum size of data that can be sent within a single work area.

Data type	Integer
Units	Bytes
Default	10000
Range	-1, 0 (no limit) and 1 to 2147483647

The following values are also used to define the maximum send size.

-1	Default.
0	No limit.

Maximum receive size

Specifies the maximum size of data that a single work area can receive.

Data type	Integer
Units	Bytes
Default	10000
Range	-1, 0 (no limit) and 1 to 2147483647

The following values are also used to define the maximum receive size.

-1	Default.
0	No limit.

Enable Web service propagation

Specifies whether the work area is propagated on Web service requests. This option is disabled by default.

Overriding work area properties

About this task

Work areas are inherently associated with the process that creates them. In the sample application, the client begins a work area and sets into it the site-identifier and priority properties. This work area is propagated to the server when the client makes a remote invocation.

Applications nest work areas in order to temporarily override properties imported from a client process. The nesting mechanism is automatic; invoking begin on the UserWorkArea interface from within the scope of an existing work area creates a nested work area that inherits the properties from the enclosing work area. Properties set into the nested work area are strictly associated with the process in which the work area was begun; the nested work area must be completed within the process that created them. If a work area is not completed by the creating process, the work-area facility terminates the work area when the process exits. After a nested work area is completed, the original view of the enclosing work area is restored. However, the view of the complete set of work areas associated with a thread cannot be decomposed by downstream processes.

Applications set properties into a work area using property modes in ensure that a particular property is fixed (not removable) or read-only (not overrideable) within the scope of the given work area.

Example

In the following code example, the server-side sample bean attempts to write directly to the imported work area; because the UserWorkArea partition is not defined to be bidirectional, this action is not permitted, and the NotOriginator exception is thrown. When the UserWorkArea partition is not defined as bidirectional, the sample bean must begin its own work area in order to override any imported properties, as shown in the second code example. If a work area in a user defined partition is used and is defined as bidirectional, this bean can set context into the work area before beginning another work area. This context set in the bidirectional case propagates back to the caller. Refer to the Work area partition service article for additional information.

```
public class SimpleSampleBeanImpl implements SessionBean {

   public String [] test() {
     ...
     String invoker = userWorkArea.getName();

     try {
       userWorkArea.set("key", "value");
     }
     catch (NotOriginator e) {
     }
     ...
  }
}
```

The following code example demonstrates beginning a nested work area, using the name of the creating class to identify the nested work area.

```
public class SimpleSampleBeanImpl implements SessionBean {

   public String [] test() {
      ...
      String invoker = userWorkArea.getName();
      try {
        userWorkArea.set("key", "value");
      }
      catch (NotOriginator e) {
      }

      // Begin a nested work area. By using the name of the creating
```

```
      // class as the name of the work area, we can avoid having
      // to explicitly set the name of the creating class in
      // the work area.
      userWorkArea.begin("SimpleSampleBean");

      ...
   }
}
```

In the example application, the client sets the site-identifier property as read-only; that guarantees that the request is always associated with the client's company identity. A server cannot override that value in a nested work area. In the following code example, the SimpleSampleBean attempts to change the value of the site-identifier property in the nested work area it created.

```
public class SimpleSampleBeanImpl implements SessionBean {

  public String [] test() {
      ...

      String invoker = userWorkArea.getName();
      try {
        userWorkArea.set("key", "value");
      }
      catch (NotOriginator e) {
      }

      // Begin a nested work area.
      userWorkArea.begin("SimpleSampleBean");

      try {
        userWorkArea.set("company",
                         SimpleSampleCompany.London_Development);
      }
      catch (NotOriginator e) {
      }
      ...
  }
}
```

retrieveAllKeys method

About this task

The UserWorkArea interface provides the retrieveAllKeys method for retrieving a list of all the keys visible from a work area. This method takes no arguments and returns an array of strings. The retrieveAllKeys method returns null if there is no work area associated with the thread. If there is an associated work area that does not contain any properties, the method returns an array of size 0.

For additional information about work area, see the com.ibm.websphere.workarea package in the API documentation. The generated API documentation is available in the information center table of contents from the path **Reference** > **APIs - Application Programming Interfaces**.

Appendix. Directory conventions

References in product information to *app_server_root*, *profile_root*, and other directories imply specific default directory locations. This topic describes the conventions in use for WebSphere Application Server.

Default product locations - IBM i

These file paths are default locations. You can install the product and other components in any directory where you have write access. You can create profiles in any valid directory where you have write access. Multiple installations of WebSphere Application Server products or components require multiple locations.

app_client_root
The default installation root directory for the Application Client for IBM WebSphere Application Server is the `/QIBM/ProdData/WebSphere/AppClient/V8/client` directory.

app_client_user_data_root
The default Application Client for IBM WebSphere Application Server user data root is the `/QIBM/UserData/WebSphere/AppClient/V8/client` directory.

app_client_profile_root
The default Application Client for IBM WebSphere Application Server profile root is the `/QIBM/UserData/WebSphere/AppClient/V8/client/profiles/`*`profile_name`* directory.

app_server_root
The default installation root directory for WebSphere Application Server - Express is the `/QIBM/ProdData/WebSphere/AppServer/V8/Express` directory.

java_home

Table 338. Root directories for supported Java Virtual Machines.

This table shows the root directories for all supported Java Virtual Machines (JVMs).

JVM	Directory
32–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/32bit`
64–bit IBM Technology for Java	`/QOpenSys/QIBM/ProdData/JavaVM/jdk60/64bit`

plugins_profile_root
The default Web Server Plug-ins profile root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver/profiles/`*profile_name* directory.

plugins_root
The default installation root directory for Web Server Plug-ins is the `/QIBM/ProdData/WebSphere/Plugins/V8/webserver` directory.

plugins_user_data_root
The default Web Server Plug-ins user data root is the `/QIBM/UserData/WebSphere/Plugins/V8/webserver` directory.

product_library
product_lib
This is the product library for the installed product. The product library for each Version 8.0 installation on the system contains the program and service program objects (similar to `.exe`, `.dll`, `.so` objects) for the installed product. The product library name is `QWAS8`*x* (where *x* is A, B, C, and so on). The product library for the first WebSphere Application Server Version 8.0 product installed on the system is `QWAS8A`. The *`app_server_root`*`/properties/product.properties` file contains the value for the product library of the installation, `was.install.library`, and is located under the *`app_server_root`* directory.

profile_root
The default directory for a profile named *profile_name* for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express/profiles/`*`profile_name`* directory.

shared_product_library
The shared product library, which contains all of the objects shared by all installations on the system, is `QWAS8`. This library contains objects such as the product definition, the subsystem description, the job description, and the job queue.

user_data_root
The default user data directory for WebSphere Application Server - Express is the `/QIBM/UserData/WebSphere/AppServer/V8/Express` directory.

The `profiles` and `profileRegistry` subdirectories are created under this directory when you install the product.

web_server_root
The default web server path is /www/*web_server_name*.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

APACHE INFORMATION. This information may include all or portions of information which IBM obtained under the terms and conditions of the Apache License Version 2.0, January 2004. The information may also consist of voluntary contributions made by many individuals to the Apache Software Foundation. For more information on the Apache Software Foundation, please see http://www.apache.org. You may obtain a copy of the Apache License at http://www.apache.org/licenses/LICENSE-2.0.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Intellectual Property & Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
USA

Trademarks and service marks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. For a current list of IBM trademarks, visit the IBM Copyright and trademark information Web site (www.ibm.com/legal/copytrade.shtml).

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Other company, product, or service names may be trademarks or service marks of others.

Index

A

active key history 1837
administrative authorization
  fine-grained security
    administrative console 1697
administrative authorization group
  fine-grained security
    administrative console 1694
administrative group roles 1681
administrative roles 1623
  authorization access 1679
administrative security
  fine-grained security 1688
    heterogeneous environments 1699
    single-server environments 1699
algorithm mapping 3391
algorithm URIs 3390
Apache Derby 344
APIs
  programmatic outbound configurations
    JSSEHelper 1773
  single sign-on 1439
applications
  security 1176
  security propagation 1678
attribute mapping
  federated repositories 1321
audit encryption keystores and certificates 1875
audit event factories
  configuration 1870
  for security auditing 1869
audit monitor 1862
audit reader
  usage 1878
audit service providers 1865
audits
  web services security
    runtime environment 3187
authentication
  configuration 1589
  default token 1548
  JAAS 1517
  Kerberos 1416
    message protection 3261
    setting up 1424
  Kerberos tokens
    web services security 3258
  LTPA tokens
    web services security 3195
  message-layer authentication 1432, 1599
  settings
    cache 1252
  single sign-on 1437
    LTPA cookies 1438
  SPNEGO 1444
    alias host name usage 1461
    HTTP requests 1449
authentication *(continued)*
  Username tokens
    web services security 3195
  users 1254
authentication cache
  configuration 1561
authentication mechanisms 1411, 1431
authentication protocol support 1596
authorization
  administrative roles 1615
  naming service 1615
  resource access 1614
  running identity mapping 1532
  technology 1615
authorization providers 1627
  built-in providers 1645
authorization roles 1621

B

bindings
  client bindings configuration 3279
  client configurations 3279
  configuration 3290
    clients 3455
    servers 3455
  provider configurations 3279
  reassigning to policy sets 2706
  server security configuration
    administrative console 3457
  STS 3271
  WS-Security
    example 3192

C

CEA 167
  administering applications 167
  cluster configuration 170
  hosting 2481, 2484
  proxy servers 2485
  servlet configuration 167
certificate expiration monitoring
  configuration 1830
certificate requests
  extraction 1813
certificate revocation list 3429
certificate signers
  clients
    auto-exchange prompt change 1821
  utilities
    signer retrieval 1820
certificate stores
  configuration
    generator bindings 3423
certificates 1796
  authority request creation 1805

certificates *(continued)*
certificate replacement 1802
configuration
example 1590, 1591
exporting 1814
extraction 1819
from certificate authorities 1813, 1817
importing 1815
replacement 1817
self-signed certificates 1801
signer certificate extraction 1818
signer certificates
adding to keystore files 1824
signer exchange 1829
collection certificate stores 3425
client configuration
administrative console 3451
configuration
consumer bindings 3432
collection certificates
configuration
cells 3433
servers 3433
commands
FileRegistryCommands 1329
signer retrieval 1730, 1740
Tivoli Access Manager configuration 1667
communications
security 1700
configuration aliases
mapping 339
configuration files
clients
ssl.client.props 1783
connection factories
automatic configuration 1497
manual configuration 1496
connection pools
settings 205, 782, 806, 833
context
caller 468, 485
context objects
fields 1850, 1858
cookies
settings 2582
CORBA naming service groups 1681
cryptographic keys
enabling
hardware devices 3441
hardware devices
web services security 3440
web services security
hardware devices 3441
cryptographic keystores
for hardware 1794
CSIV2 1596
client configuration 1584
configuration 1568
inbound communications 1562
custom login migration
CustomLoginServlet class 1167
custom object pools
settings 1030
custom properties
federated repositories 1321
settings 297
web services security 3174, 3176

D

data access
administering applications 201
data sources
configuration 1373
settings 284
databases
Apache Derby 344
settings 2592
default bindings
cells 2705
servers 2694, 2705
default collection certificate stores
configuration
administrative console 3453
default policy sets
bindings 2701
Version 6.1 bindings 2704
delegations 1643
directory
installation
conventions 200, 204, 242, 384, 388, 3581
distributed cache
synchronous update 3210
token recovery 3210
web services security
administrative console 3255
dynamic annotations
servlet security 1642
dynamic cache service
administering 349
settings 351
usage 349
dynamic group support
directory servers 1282
Tivoli Directory Server 1283
dynamic groups
LDAP 1408
nested groups 1408
dynamic member attributes 1405
federated repositories 1404
dynamic roles
caching properties 1655

E

EJB security
authentication protocol 1593
emitters
interfaces
base generic emitters 1867, 1869
encryption information 3357

endpoint security
  configuration management 1778
endpoint URL information
  configuration
    for HTTP bindings 2609
    JMS bindings 2611
  HTTP 2610
enterprise identity mapping 1440
  configuration 1490
  connection factories 1493
  deployment 1499
  prerequisite applications 1489
Enterprise JavaBeans (EJB)
  container interoperability 401
  container system properties 393
  containers 391
  deployment 385
    overview 386
  enterprise beans
    deployment 383
  managing containers 390
  settings 383
    containers 392
  troubleshooting
    EJBDEPLOY relationships 388
entity beans
  administering 389
entity types 1378
event type filters 1856

F

federated repository wizard 1327
FileRegistrySample.java 1293
files
  rrdSecurity.props file 3438
  SAMLIssuerConfig.properties file 3285

G

generic events
  interface
    example 1857
global security
  custom properties 1217
    adding 1216
    deleting 1218
group attribute definitions
  configurations 1399

H

HTTP basic authentication 3174
HTTP session management
  custom properties 2583
  settings 2577
HTTP sessions
  recovering 2579
  tracking configuration 2579
HTTP SSL configuration 3172

I

IBM Optim pureQuery Runtime 442
identity assertions
  configuration 1589
  downstream servers 1597
  trust validation 1598
identity mapping
  custom login module 1538
  inbound configuration 1533
  outbound configuration 1540
inbound transports
  configuration 1575
interfaces
  generic event factories
    example 1871
interoperating
  previous versions 1160

J

JAAS 1516
  application login customization 1530
  custom login module development 1521
  identify assertion enablement 1531
  Kerberos login modules
    system updates 3268
  web authentication 1519
JACC 1628
  ContextID format 1631
  external provider enablement 1649
  interface support 1662
  policy context handlers 1631
  policy propagation 1632
  provider implementation class registration 1633
  providers 1630
JAR files
  configuration 1494
  shared libraries 1495
Java 2 security 1176
  access control exception 1185
  policy files 1181
Java Servlet 3.0
  security support 1640
Java thin clients
  migration 1172
JAX-RPC
  administration
    message-level security 3293
  consumer binding
    applications 3310
  encryption methods
    message confidentiality protection 3355, 3378
  generator signing configuration
    message integrity 3294
  key information
    consumer bindings 3377
    generator bindings 3375
  key information configuration
    applications 3314

JAX-RPC *(continued)*
key locators
cell 3414
configuration 3405
consumer bindings 3412
server 3414
message authenticity protection
token consumers 3345
message integrity protection
consumer signing 3309
message-level security
cells 3369
for applications 3294
servers 3369
secure messages
request consumers 3294
request generators 3293
response consumers 3294
response generators 3293
signing information
consumer bindings 3372
generator bindings 3295, 3369
token consumers
message authenticity protection 3393
token generators
message authenticity protection 3382
web services security
configuration 3402
JAX-RPC web services
HTTP basic authentication
administrative console 3173
JAX-RS
deployment 3159
planning 3145
RESTful services 3145
JAX-WS
administration
message-level security 3187
application deployment model 2600
asynchronous response servlet usage 3475
default bindings
web services security 3292
invocation
HTTP transports 3475
Kerberos token
policy sets 3259
listeners
JMS asynchronous response message 3478
third-party engine 2602
web services invocation
SAOP over JMS transports 3477
JDBC
provider configuration 273
provider settings 275
JMX
scheduler configuration 1138
JNDI
configuration
bindings 1015
JPA
administering 431
JPA *(continued)*
configuration 432
default settings 439
JSF files
deployment 2517
JSP engine 2558
JSP engine parameters
configuration 2556
JSP files
backing up 2571
deployment 2517
recovering 2571
troubleshooting 2567

K

key
locators 3407
key generation classes
development
example 1842
key generation retrieval
from key set groups 1841
key information 3317
consumer bindings
applications 3325
key information references 3312
key management
cryptographic usage 1834
key managers 1799
X.509 certificate identities 1711
key set groups 1845
configuration creation 1840
key sets 1838
configuration creation 1835
keys 3339
locator configuration
administrative console 3454
keystore configurations
preexisting keystore files 1792
remote management 1795
keystore files 1796
signer exchange 1829
keystores
internal password records
recreating .kdb files 1793

L

LDAP
adding users 1278
advanced LDAP settings 1268
bindings 1286
directory servers 1275
key set groups 1414
key sets 1414
performance 1383
search filters 1272
security failover 1409
settings 1343
stand-alone registry settings 1264

LDAP *(continued)*
user group memberships 1279
LDAP entity types 1397
federated repositories 1396
listeners
endpoints
overview 2623
services
overview 2623
local operating system wizard 1260
login configuration
WSLogin 1542
login mappings 3441
LTPA 1413
keys 1415
single sign-on 1438
enablement 3439

M

mail providers
configuration 499
settings 501
mail sessions
configuration 503
mappings
ports 2607
member attributes 1402
federated repository 1401
messages
asynchronous response listener usage 3476
authenticity protection
for applications 3327
endpoint management 250
inbound configuration 1582
outbound configuration 1583
methods
encryption
message confidentiality protection 3367, 3378, 3380

N

name servers
configuration 1023
namespace bindings 1017
naming
settings
binding types 1017
CORBA object bindings 1021
EJB bindings 1020
indirect lookup bindings 1022
string bindings 1019
naming roles
user assignment 1683
naming servers
settings 1023
nested group support
directory servers 1282
Tivoli Directory Server 1283
new administrative authorization groups 1693, 1694
nonce
cache distribution 3405
configuration 3404
applications 3448
servers 3447
web services security tokens 3447
notifications 1833
security audit subsystem failures 1861

O

object pool managers 1026
MBeans 1031
object pool services
settings 1031
object pools
MBeans 1031
resources for learning 1031
settings 1028
usage 1025
objects
caching properties 1656
ORB
administering 1033
custom properties 1036
Java
character code set conversions 1045
settings
services 1033
outbound transports
configuration 1579

P

part references 3304
personal certificate requests 1811
personal certificates 1807
policies
configuration
custom properties 2755
WS-Addressing 2719
WS-Security 2738
Java 2 security migration 1168
web services 2758
policy sets
attaching to service artifacts 2710
bindings 2707
default policy set 2697
disabling policies 2757
enabling policies
administrative console 2757
importing 2677
Kerberos
sample bindings 3268
STS 3271
system 3224
system configuration
administrative console 3227
system definition
administrative console 3228

policy sets *(continued)*
trust service
request protection 3196
web services security 3191
WS-Security
example 3192
port retrieval 1823
portlet container
settings 1123
portlet fragment caching
configuration 362
wsadmin tool 365
principal mapping
global sign-on configuration 1512
profiles
certificate options 1722
programmatic login migration 1164
properties
rrdSecurity.props file 3438
SAMLIssuerConfig.properties file 3285
web services security 3411
properties files
group.props file 1310
users.props file 1310
property extensions
DB2 1371
protection wizard 1193
protocol providers
settings 502
proxy servers
third-party HTTP reverse servers 1433

R

RAR
bean validation 228
installation 180, 182
realms
configuration settings 1319
security 1186
registries
custom user registry development 1311
federated repositories 1394
LDAP 1261, 1286
stand-alone LDAP 1407
user group memberships 1279
local operating systems 1257, 1258
selecting 1254
Remote request dispatcher 2572, 2575
reports
security configuration 1213
repositories 1381
federated 1318, 1325, 1330
changing passwords 1324
configurations 1380
custom adapters 1386, 1387, 1388, 1390
custom properties 1321
entity types 1377
entry mapping 1374
external repositories 1352
file details 1322
repositories *(continued)*
federated *(continued)*
LDAP 1333, 1334, 1335, 1337, 1339
limitations 1323
performance 1382
property extensions 1353, 1366
realm management 1316
user registries 1392
LDAP 1331
replication 1381
selecting 1254
request receiver bindings 3461
request sender bindings 3460
resource adapter archive
installation 180
resource adapters
configuration 237
installation 181
resource environment entries 511
administering 508
configuration 508
resource environment providers 509
settings 509
response receiver bindings 3463
response sender bindings 3463
RESTful applications
HTTP headers 3151
HTTP response codes 3151
media types definitions 3152
parameter definitions 3155
resource definitions 3146
resource methods definition 3149
URI pattern definitions 3147
role-based policy framework 1656
RRD 2572, 2575
RRD applications
administering 2572, 2575

S

SAML
message-level tokens 3269
secure messages 3269
token propagation 3163
tokens
attribute creation 3167
client binding configuration 3274
client bindings configuration 3286
provider binding configuration 3274, 3286
wsadmin commands 3289
user attributes 3168
SAML applications
deployment 3163
SAS
client configuration 1584
scheduler daemon 1130
schedulers
calendars 1127
configuration 1132
creating databases 1141

schedulers *(continued)*
creating tables
administrative console 1148
using DDL files 1152
using scripting 1149
using the administrative console 1148
managing 1130
settings 1136
table definition 1146
security
administrative 1174
applications 1176
coexisting 1159
custom properties 1194, 1211
domains 1245, 1252
enabling 1159, 1172
interoperating 1159
migrating 1159
multiple domains
copying 1241
creating 1238
deleting 1241
inbound trusted realms 1245
realm names 1251
realms 1186, 1251
settings
add key alias references 1837
add signer certificates 1825
administrative user password 1326
administrative user roles 1679
advanced LDAP 1268
audit event factories 1870
audit notifications 1863
audit record encryption 1876
audit record keystore files 1878
audit record signing configurations 1877
audit service providers 1865
authentication cache 1252
authentication protocol for client configurations 1585
certificate expiration management 1831
certificate requests 1806
CSIV2 communications 1561
CSIV2 inbound communications 1564
CSIV2 outbound communications 1569
CSIV2 outbound transport communications 1580
CSIV2 transport inbound communications 1577
custom properties 1212
dynamic member attributes 1406
dynamic outbound endpoint SSL configuration 1780
entity types 1379
entry mapping repository 1376
event type filters 1855
external authorization providers 1645
global security 1188
group attribute definitions 1399, 1400
JACC providers 1646
key managers 1800
key set groups 1845
key sets 1838
security *(continued)*
settings *(continued)*
keystore files 1797
LDAP entity types 1398
local operating system 1259
member attributes 1403
naming service users 1679
notifications 1833
personal certificate requests 1811
property extension repositories 1357
quality of protection 1781
reference 1382
SAS authentication protocol 1589
SAS inbound transport communications 1578
SAS outbound transport communications 1582
self-signed certificates 1808
signer certificates 1826
single sign-on 1501
SSL 1747
stand-alone custom registries 1290
stand-alone LDAP registry 1264
Tivoli Access Manager JACC providers 1652
trust and key managers 1762
trust association interceptors 1437
trust associations 1436
trust managers 1764
setup 1159
testing 1212
security annotations 1638
security attributes
default authorization token 1551
default propagation token 1554
default single sign-on 1560
propagation 1544, 1549
security audit data
protection 1872
security auditing
context objects 1858
security auditing subsystems
enablement 1849
security audits 1849
default service provider configuration 1864
event type filter creation 1854
events 1854
infrastructure 1847
record encryption 1873
records signing 1874
third-party service providers
configuration 1868
Security Configuration wizard 1213
security domains 1222
configuration 1219
service endpoints
attachments
administrative console 3235
servlet caching
configuration 361
sesions
configuration 499
signer certificates 1825
exchange 1828

signer certificates *(continued)*
extraction 1819
signing information 3298
single sign-on
configuration 1510
RMI_OUTBOUND 1543
Tivoli Access Manager 1500
trust associations 1509
enterprise identity mapping 1487
HTTP requests 1444
principle mapping 1441
web user authentication 1441
SIP
administering applications 2439
container configuration 2439
custom properties 2440, 2466
deployment 2437
wsadmin scripting 2438
DNS 2453
failover 2461
routers 2464, 2479
configuring 2462
settings 2465, 2466
settings 2454, 2465, 2466
stack settings 2457
startup order 2466
timers 2458, 2460
SOAP
secure conversation 3198
SQLJ profiles 194
SSL 1700, 1747
alias selection 1775
application server 1719
CA client creation 1790
central management 1718
certificate expiration monitoring 1732
certificate management 1736, 1745
certificates
iKeyman usage 1735
client authentication enablement 1777
client certificate authentication 1748
cluster isolation 1719
configuration creation 1741
configurations 1706
custom key manager creation 1766
custom trust decisions
custom trust managers 1765
custom trust managers 1761
default chained certificates 1724
dynamic configuration associations 1771
dynamic configuration updates 1735
dynamic inbound and outbound endpoints 1779
dynamic outbound selections 1717
inbound endpoints 1777
inbound scope associations 1774
key management 1745
keystore configurations 1715
nodes 1719
outbound scope associations 1774
programmatic outbound configurations 1773
SSL *(continued)*
remote ports
signer retrieval 1822
replacement certificates
cells 1804
nodes 1803
scopes 1746
secure installations 1728
stand-alone custom registries 1289
configuration 1287
Stand-alone custom registry wizard 1292
Stand-alone LDAP registry wizard 1267
startup beans 2487
enabling 2488
startup beans services
settings 2489
static roles
caching properties 1654
system policy sets 3229
system properties
Enterprise JavaBeans (EJB) 393
system-dependent configuration 1657

T

targets
trust service
administrative console 3249
trust service endpoint configuration
administrative console 3248
TCP/IP transports
virtual private network
example 1592
timer managers
configuration 7
settings 9
timer service
configuration 402
Tivoli Access Manager 1636
administrative role changes
propagation 1684
administrative user creation 1651
authentication
node migration 1171
authorization server configuration 1659
configuration
web servers 1509
embedded enablement 1666
administrative console 1677
group configuration 1658
JACC 1652
provider configuration properties 1654
JACC provider enablement 1665
JACC providers 1634
administrative console 1649
unconfiguration 1678
Java EE resource access 1644
logs 1660, 1661
role-based security 1633
security roles 1658
security users 1658

Tivoli Access Manager *(continued)*
single sign-on 1500, 1503, 1504
trusted user accounts 1507
utilities
EAR file migration 1685
Tivoli Directory Server
group support 1283
token consumers 3397
token generators 3386
token providers
security context
administrative console 3241, 3242
tokens
consumer configuration
secure conversation 3212
derived key 3204
generator configuration
secure conversation 3212
pluggable configuration
administrative console 3464
security context 3216
disabling submission draft levels 3244
transforms 3306
trust anchors 3418
configuration
cells 3422
consumer bindings 3420
generator bindings 3416
servers 3422
trust association interceptors 1436
trust association migration 1162
trust associations 1433
trust managers 1763
X.509 certificate identities 1709
trust service 3214
attachment configuration
administrative console 3233
attachments 3237
targets 3251
token custom properties 3243
token providers 3247
trusted IDs
configuration
cells 3435
servers 3435
evaluators 3436

W

web applications
administering 2527
settings
asynchronous request dispatching 2573, 2574
web container
custom properties 2530
settings 2528
troubleshooting 2551
web services
administering
deployed applications 2622
client bindings 2606
web services *(continued)*
client bindings configuration 2605
clients
port information 2608
deployment 2596
planning 2595
secure conversation
standard 3212
secure conversations 3199, 3201
client cache 3202
settings
options to perform web services deployment 2597
publish WSDL compressed files 2616
trust service
configurations 3202
trust standard 3225
web services applications
application server deployment 2596
deployment
for clients 2604
web services security
administration 3171
binding configuration 3290
client security bindings 3342
configuration
JAX-RPC 3402
cryptographic device enablement 3440
default policy sets 2673
hardware cryptographic devices
configuration 3440
HTTP outbound transport communications
administrative console 3171
Java properties 3172
mixed cluster environment 3205
properties 3411
runtime configuration updates 3253
secure conversations
distributed cache 3206
security context tokens 3208
session affinity 3206
security context tokens
reliable messaging 3209
server security bindings 3343
server-side collection certificates
administrative console 3452
settings
algorithm mapping configurations 3392
algorithm URI configuration 3390
callback handler 3336
certificate revocation list configurations 3430
collection certificate store configurations 3426
encryption information configuration 3358, 3363
JAAS configuration 3353
key configuration 3340
key information configuration 3318
key information reference configuration 3313
key locator configurations 3408
login bindings configuration 3458
part reference configuration 3305

web services security *(continued)*
settings *(continued)*
request consumer (receiver) binding configuration 3350
request generator (sender) binding configuration 3332
response consumer (receiver) binding configuration 3351
response generator (sender) binding configuration 3334
security cache 3256
signing information configuration 3299
system policy sets 3231
token consumers 3398
token generator configurations 3387
transforms 3307
trust anchors 3419
trust service attachments 3240
trust service targets 3253
trust service token providers 3244
trusted ID evaluator configurations 3437
web services runtime updates 3254
web services security property configurations 3411
X.509 certificate configurations 3428
WebSEAL
configuration 1508
single sign-on 1500
work manager
settings 12
wsadmin commands
repository setup 1359
self-issued SAML tokens 3289
web services deployment
wsdeploy 2599
wsdeploy 2599
WSDL
file publication 2615
using URLs 2617

X

x.509 certificates 3428
XML digital signatures
configuration
Version 5.x web services 3441
XML encryption
configuration
Version 5.x web services 3458